

A MODEL FOR VULNERABILITY FORECASTING

BY

HEIN S. VENTER



A MODEL FOR VULNERABILITY FORECASTING

by

HEIN S. VENTER

THESIS

submitted in fulfilment of the
requirements for the degree

DOCTOR OF PHILOSOPHY

in the subject

COMPUTER SCIENCE



FACULTY OF NATURAL SCIENCES

UNIVERSITY
OF
JOHANNESBURG
of the

RAND AFRIKAANS UNIVERSITY

PROMOTER: PROFESSOR JAN H. P. ELOFF

MAY 2003

ABSTRACT

Internet and network security forms an interesting and topical, yet challenging and developing research domain. In this domain, a taxonomy of information security technologies is identified. This taxonomy is divided into two mainline entities, namely proactive and reactive information security technologies. This thesis is specifically concerned with proactive information security technologies, the focus being on a specific proactive information security technology – vulnerability scanning.

Vulnerability scanning is implemented by vulnerability scanner (VS) products. VS products are used proactively to conduct vulnerability scans to identify vulnerabilities so that they can be rectified before they can be exploited by hackers. However, there are currently many problems with state-of-the-art VS products. For example, a vulnerability scan is time-consuming and a vast number of system resources are occupied, leading to the degradation of network and system performance. Furthermore, VS products lack the intelligence that is required to deal with new vulnerabilities that appear like clockwork. Current VS products also differ extensively in the way that they can detect vulnerabilities, as well as in the number of vulnerabilities that they can detect.

These problems motivated the researcher to create a model for vulnerability forecasting (VF). The uniqueness of the VF model lies in its holistic approach to addressing these problems while maintaining its end goal – that of being able to do a vulnerability forecast of how vulnerabilities will occur in the near future. Such a vulnerability forecast would, therefore, enable an organisation to use it proactively as part of a risk management scheme.

Furthermore, in order to demonstrate the feasibility of implementing the proposed model, a report on the development of a prototype for vulnerability forecasting is included. Rather than reinventing the wheel, the prototype incorporates the use of current state-of-the-art VS products in its VF process. This is advantageous in the sense that the prototype is independent of a specific VS product. It is because of the

latter that a standardisation technique had to be used to refer to vulnerabilities in the same way since different VS products do not refer to and detect similar vulnerabilities in the same way. This standardisation technique introduced in this thesis is known as harmonising vulnerability categories.

This thesis contributes to the understanding of vulnerability scanning techniques and how vulnerability scanning can be utilised more effectively by doing vulnerability forecasting. The thesis also paves the way for numerous potential future research projects in the domain of Internet and network security.



AFRIKAANSE OORSIG

Internet- en netwerksekuriteit vorm 'n interessante en aktuele, dog uitdagende en ontwikkelende navorsingsgebied. Op hierdie gebied word 'n taksonomie van inligtingsekuriteitstechnologieë geïdentifiseer. Hierdie taksonomie word in twee hooflyntiteite verdeel, naamlik proaktiewe en reaktiewe inligtingsekuriteitstechnologieë. Hierdie proefskrif handel spesifiek oor proaktiewe sekuriteitstechnologieë en die fokus is op 'n spesifieke proaktiewe inligtingsekuriteitstechnologie – kwesbaarheidsaftasting.

Kwesbaarheidsaftasting word deur kwesbaarheidsafts- (VS-) produkte geïmplementeer. VS-produkte word proaktief gebruik om kwesbaarheidsaftastings uit te voer om kwesbaarhede te identifiseer sodat hulle reggestel kan word voordat hulle deur krakers uitgebuit word. Tans is daar egter baie probleme met die nuutste VS-produkte. 'n Kwesbaarheidsaftasting is byvoorbeeld tydrowend en 'n groot hoeveelheid stelselhulpbronne word in beslag geneem, wat tot die verlaging van netwerk- en stelselprestasie lei. Verder beskik VS-produkte nie oor die nodige intelligensie om nuwe kwesbaarhede wat klokslag verskyn, te hanteer nie. Huidige VS-produkte verskil ook hemelsbreed wat betref die manier waarop hulle kwesbaarhede opspoor sowel as die getal kwesbaarhede wat hulle kan opspoor.

Hierdie probleme het die navorser gemotiveer om 'n model vir kwesbaarheidsvoorspelling (VF) te skep. Die uniekheid van die VF-model lê in sy holistiese benadering tot die aanspreek van hierdie probleme terwyl die einddoel – om te kan voorspel hoe kwesbaarhede in die nabye toekoms sal voorkom – gehandhaaf word. So 'n kwesbaarheidsvoorspelling sal 'n organisasie dus in staat stel om dit proaktief te gebruik as deel van 'n risikobestuursplan.

Verder, om die uitvoerbaarheid van die implementering van die voorgestelde model aan te toon, word 'n verslag oor die ontwikkeling van 'n prototipe vir kwesbaarheidsvoorspelling ingesluit. In plaas daarvan om weer die wiel uit te vind, inkorporeer die prototipe die gebruik van die heel nuutste VS-produkte in sy VF-proses. Dit is voordelig in dié sin dat die prototipe onafhanklik is van 'n spesifieke

VS-produk. Vanweë laasgenoemde moes 'n standaardiseringstegniek gebruik word om op dieselfde manier na kwesbaarhede te verwys, aangesien verskillende VS-produkte nie op dieselfde manier na kwesbaarhede verwys of hulle op dieselfde manier opspoor nie. Hierdie standaardiseringstegniek wat in die proefskrif bekend gestel word, staan bekend as die harmoniëring van kwesbaarheidskategorieë.

Hierdie proefskrif dra by tot die begrip van kwesbaarheidsaftastegnieke en hoe kwesbaarheidsaftasting meer effektief benut kan word deur kwesbaarheidsvoorspellings te doen. Hierdie proefskrif baan ook die weg vir talle potensiële toekomstige navorsingsprojekte op die gebied van Internet- en netwerksekuriteit.



ACKNOWLEDGEMENTS

“Life is a journey, enjoy the ride” is a philosophy that many people live by. This philosophy is applicable to the essence of this thesis as well. In the journey of the thesis, the ride was sometimes bumpy, sometimes flat, and sometimes, alas... completely off-road! Like all journeys, this one has also come to an end. The journey would not have been possible without an excellent vehicle, and that vehicle would embody everyone who made it possible for this research project to be completed. It is therefore appropriate and absolutely crucial to convey my sincere appreciation to all who made the journey possible. Without your support, guidance and encouragement, it would have been impossible.

A special word of thanks to:

- God, my Anchor and my Reason for being, for the talents and energy He gave me to complete this research.
- My fiancée, René, for her undivided love, support and encouragement amidst difficult times.
- My parents, for their continuous support and for affording me a proper education in financially challenging times.
- My promoter, Professor Jan Eloff, for his peerless guidance and continuous encouragement.
- All my friends, who continuously encouraged me and for believing in me. A special word of thanks to Damian Cholewka, Keith Snyman and Gert van Rensburg for listening to my frustrations.
- Pierre Visser, for his contribution to the development of the vulnerability forecasting prototype.
- My colleagues at the University of Pretoria, for assisting me wherever possible. A special word of thanks to Madel Morkel for always being eager to help with the administration regarding the thesis.
- My former colleagues at the Rand Afrikaans University for their support through the years. A special word of thanks to Mrs Ina Erasmus and Mrs Naomi Strijdom for always being friendly and available to help with all the administration regarding the thesis.

- Glenda Buncombe, for editing the thesis professionally and promptly.
- The National Research Foundation for financial assistance.
- Anyone else who contributed – even if in the faintest manner – to the completion of this research, I thank you all.



CONTENTS

ABSTRACT	v
AFRIKAANSE OORSIG	vii
ACKNOWLEDGEMENTS	ix
1 INTRODUCTION	1
1.1 INTRODUCTION	1
1.2 MOTIVATION FOR THIS STUDY.....	1
1.3 PROBLEM STATEMENT.....	5
1.4 TERMINOLOGY USED IN THIS THESIS.....	8
1.4.1 Information security	8
1.4.2 Intrusion detection	9
1.4.3 Vulnerability scanning	9
1.4.4 Risk management	10
1.5 THESIS LAY-OUT	10
2. A TAXONOMY FOR INFORMATION SECURITY TECHNOLOGIES	13
2.1 INTRODUCTION	13
2.2 A TAXONOMY FOR INFORMATION SECURITY TECHNOLOGIES.....	13
2.2.1 Proactive information security technologies	16
2.2.1.1 Cryptography	16
2.2.1.2 Digital signatures	17
2.2.1.3 Digital certificates	17
2.2.1.4 Virtual private networks.....	18
2.2.1.5 Vulnerability scanners.....	18
2.2.1.6 Anti-virus scanners.....	19
2.2.1.7 Security protocols	19
2.2.1.8 Security hardware	20

2.2.1.9	Security SDKs.....	20
2.2.2	Reactive information security technologies	21
2.2.2.1	Firewalls	21
2.2.2.2	Access control.....	22
2.2.2.3	Passwords.....	22
2.2.2.4	Biometrics.....	23
2.2.2.5	Intrusion detection systems	23
2.2.2.6	Logging.....	24
2.2.2.7	Remote accessing.....	24
2.3	CONCLUSION	25
3.	STATE-OF-THE-ART INTRUSION DETECTION AND VULNERABILITY	
	SCANNING	27
3.1	INTRODUCTION	27
3.2	INTRUSION DETECTION.....	28
3.2.1	What is intrusion detection?	28
3.2.2	The architecture of IDSs.....	28
3.2.2.1	Pattern-matching IDS architecture.....	30
3.2.2.2	Anomaly detection IDS architecture	32
3.2.3	Other approaches to IDS architectures	33
3.2.3.1	IDML-based intrusion detection	33
3.2.3.2	An IDS architecture for detecting TCP SYN flooding	35
3.2.4	Commercially available IDSs.....	37
3.2.5	The problems with IDSs.....	37
3.3	VULNERABILITY SCANNING	37
3.3.1	What is vulnerability scanning?	37
3.3.2	The architecture of VSs.....	38
3.3.3	Another approach to VS architectures	42
3.3.4	Commercially available VSs	43
3.3.5	The problems with VSs.....	44
3.4	CONCLUSION	45
4.	HARMONISING VULNERABILITY CATEGORIES	47
4.1	INTRODUCTION	47

4.2	METHOD OF IDENTIFYING CATEGORIES.....	47
4.3	HARMONISED VULNERABILITY CATEGORIES	48
4.3.1.	Password cracking and sniffing.....	49
4.3.2	Network and system information gathering	50
4.3.3	User enumeration and information gathering	50
4.3.4	Backdoors, Trojans and remote controlling	51
4.3.5	Unauthorised access to remote connections and services	52
4.3.6	Privilege and user escalation.....	52
4.3.7	Spoofing or masquerading	53
4.3.8	Misconfigurations.....	54
4.3.9	Denial-of-services (DoS) and buffer overflows	54
4.3.10	Viruses and worms	55
4.3.11	Hardware specific.....	55
4.3.12	Software specific and updates	56
4.3.13	Security policy violations	57
4.4	STANDARDISATION OF VULNERABILITIES	58
4.5	CONCLUSION	58
5.	VULNERABILITY SCANNER PRODUCTS	61
5.1	INTRODUCTION	61
5.2	VS PRODUCTS.....	61
5.2.1	VS products overview.....	62
5.2.2	CyberCop Scanner	62
5.2.2.1	Practical experience with the CyberCop Scanner	62
5.2.2.2	CyberCop Scanner vulnerability database.....	63
5.2.3	Cisco Secure Scanner.....	63
5.2.3.1	Practical experience with the Cisco Secure Scanner	64
5.2.3.2	Cisco Secure Scanner vulnerability database.....	65
5.2.4	SAINT	66
5.2.4.1	Practical experience with the SAINT	66
5.2.4.2	SAINT vulnerability database.....	66
5.2.5	Internet Security Scanner (ISS).....	66
5.2.5.1	Practical experience with the ISS	67
5.2.5.2	ISS vulnerability database.....	68

5.2.6	Nessus Security Scanner	69
5.2.6.1	Practical experience with the Nessus Security Scanner.....	69
5.2.6.2	Nessus Security Scanner vulnerability database.....	70
5.3	SUMMARY OF CURRENT VS PRODUCTS	71
5.3.1	Mapping onto harmonised vulnerability categories	72
5.3.2	Differences in VS products	73
5.3.2.1	2: Network and system information gathering	73
5.3.2.2	4: Backdoors, Trojans and remote controlling.....	74
5.3.2.3	8: Misconfigurations.....	75
5.3.2.4	9: Denial-of-service (DoS) and buffer overflows.....	76
5.3.2.5	13: Security policy violations.....	78
5.4	CONCLUSION	79
6.	VULNERABILITY FORECASTING – A CONCEPTUAL MODEL	81
6.1	INTRODUCTION	81
6.2	PROBLEMS WITH STATE-OF-THE-ART VSS.....	81
6.3	CONCEPT OF VULNERABILITY FORECASTING	84
6.3.1	Defining the term “vulnerability forecasting”	84
6.3.2	A conceptual model for VF.....	84
6.3.2.1	Level 1 of the conceptual VF model	85
6.3.2.2	Level 2 of the conceptual model	85
6.3.2.2.1	<i>VS technology (current).....</i>	<i>86</i>
6.3.2.2.2	<i>Vulnerability harmonisation.....</i>	<i>92</i>
6.3.2.2.3	<i>Vulnerability forecasting.....</i>	<i>98</i>
6.3.2.2.4	<i>Merging the subcomponents for the conceptual VF model.....</i>	<i>102</i>
6.4	CONCLUSION	103
7.	THE VULNERABILITY FORECAST ENGINE	105
7.1	INTRODUCTION	105
7.2	INPUT TO THE VF ENGINE	105
7.3	EXPLANATION OF THE VF TECHNIQUE	107
7.3.1	Using FEIs for VF	108
7.3.1.1	Step 1: Determine fuzzy groups for a vulnerability forecast	109
7.3.1.2	Step 2: Defuzzify “fuzzy” vulnerabilities	110
7.3.1.3	Step 3: Define and calculate the membership function.....	112

7.3.1.4	Step 4: Defuzzify “fuzzy” scans.....	113
7.3.1.5	Step 5: Calculate the maximum over the minima and the FEL.....	114
7.3.2	FEI for each harmonised vulnerability category.....	115
7.4	CONCLUSION	116
8.	A PROTOTYPE FOR VULNERABILITY FORECASTING	117
8.1	INTRODUCTION	117
8.1.1	The aim of the prototype	117
8.1.2	The VF model and the prototype	118
8.2	DEVELOPMENT OF THE VF PROTOTYPE.....	120
8.3	INSTALLATION OF THE VF PROTOTYPE.....	121
8.4	OPERATION OF THE VF PROTOTYPE.....	121
8.4.1	Background to the VF Prototype	121
8.4.2	The scan scenario	123
8.4.2.1	The platform.....	123
8.4.2.2	The scenario.....	123
8.4.3	Setting up the VF Prototype parameters.....	125
8.4.3.1	Setting up the Adjective List.....	127
8.4.3.2	Setting up the Harmonised Vulnerability Categories	128
8.4.3.3	Setting up the VS product used.....	129
8.4.3.3.1	Specifying the VS product name.....	129
8.4.3.3.2	Specifying the vulnerability database path	129
8.4.3.3.3	Specifying the tables used.....	129
8.4.3.3.4	Specifying the software package categories.....	130
8.4.3.3.5	Specifying the vulnerability mapping.....	131
8.4.3.3.6	Specifying the format of the history scan data.....	132
8.4.4	Using the VF Prototype software	133
8.4.4.1	Analysing the history scan data.....	134
8.4.4.2	Performing a vulnerability forecast	136
8.4.4.2.1	Compiling the fuzzy groups.....	138
8.4.4.2.2	Compiling the mapping table.....	140
8.4.4.2.3	Compiling the membership function.....	141
8.4.4.2.4	Viewing calculations.....	142
8.4.4.3	Validating a vulnerability forecast	145
8.5	CONCLUSION	147

9. CONCLUSION	149
9.1 INTRODUCTION	149
9.2 REVISITING THE PROBLEM STATEMENT	149
9.3 FUTURE RESEARCH	152
9.4 EPILOGUE	153

APPENDICES

A	INSTALLING THE VF PROTOTYPE SOFTWARE AND ADDITIONAL SOFTWARE COMPONENTS	155
B	SOURCE CODE OF THE VF PROTOTYPE	161
C	CYBERCOP SCANNER REPORT	299
D	THE CYBERCOP SCANNER VULNERABILITY DATABASE	309
E	VULNERABILITY HISTORY DATA	395
F	PAPERS PUBLISHED	421



LIST OF FIGURES

2.1	A taxonomy of information security technologies	14
3.1	The typical location of an IDS in a network.....	29
3.2	Pattern -matching IDS architecture	31
3.3	Anomaly detection IDS architecture	32
3.4	An architecture for intrusion detection based on IDML.....	33
3.5	A typical finite intrusion pattern state machine	35
3.6	An intrusion detection architecture for detecting TCP SYN flooding	36
3.7	The location of a VS in a network.....	38
3.8	An architecture for a VS	41
3.9	A distributed architecture for vulnerability scanning.....	42
5.1	An extract from the CyberCop Scanner report	63
5.2	An extract from the Cisco Secure Scanner report.....	65
5.3	An extract from the ISS report.....	68
5.4	Vulnerability mapping of different VS products onto the harmonised vulnerability categories.....	72
5.5	Number of vulnerabilities scanned for by different VS products for harmonised vulnerability category 2: network and system information gathering.....	73
5.6	Number of vulnerabilities scanned for by different VS products for harmonised vulnerability category 4: backdoors, Trojans, and remote controlling	75
5.7	Number of vulnerabilities scanned for by different VS products for harmonised vulnerability category 8: misconfigurations	76
5.8	Number of vulnerabilities scanned for by different VS products for harmonised vulnerability category 9: denial-of-services (DoS) and buffer overflows	77
5.9	Number of vulnerabilities scanned for by different VS products for harmonised vulnerability category 13: security policy violations	78

6.1	Level 1 – A conceptual model for vulnerability forecasting	85
6.2	The VS technology (current) component.....	86
6.3	VF logical database part 1: vulnerability data.....	87
6.4	Vulnerability data report.....	88
6.5	VF logical database part 2 added: scan data.....	89
6.6	Scan data report for scan 1	90
6.7	The scan result report: vulnerabilities found on different hosts on a network during a specific scan.....	92
6.9	VF logical database part 3 added: harmonised vulnerability category data....	94
6.10	VS product vulnerabilities mapped on to the harmonised vulnerability categories	95
6.11	Scan result report with vulnerabilities mapped onto the harmonised vulnerability categories.....	96
6.12	VF logical database part 4 added: harmonised history data	97
6.13	The vulnerability forecast component.....	98
6.14	VF logical database part 5 added: forecast history data.....	99
6.15	A vulnerability forecast result report for vulnerability forecasts 1 to n	101
6.16	The conceptual model for vulnerability forecasting	103
7.1	The harmonised history data for m scans	106
7.2	The 5 steps for determining the FEI for each harmonised vulnerability category	109
7.3	Membership function	112
8.1	The VF Prototype's scope in terms of the VF model as indicated by the dark black line.....	118
8.2	The VF Prototype's scope in terms of the VF model as indicated by the dark black line.....	119
8.3	Relational schema of the database implementation in the VF Prototype	120
8.4	CyberCop Scanner network scan scenario	124
8.5	Vulnerabilities uncovered by CyberCop Scanner during scan 1 for each CyberCop Scanner vulnerability category.....	124
8.6	Running the VF Prototype software	125
8.7	The Vulnerability Forecasting (VF) Prototype main window.....	126

8.8	The VF Prototype – Options window.....	127
8.9	The Adding Adjective input box.....	127
8.10	The VF Prototype – Software Package Setup window.....	128
8.11	Software Package Setup window: Harmonised mapping	130
8.12	Specifying the scan data file structure of CyberCop Scanner.....	132
8.13	The VF Prototype – Options window filled in	133
8.14	Loading the history scan data.....	134
8.15	The history scan data loaded and mapped.....	135
8.16	Graph showing information about Scan 15	136
8.17	Showing the mapped data for the harmonised vulnerability categories	137
8.18	The first three steps for doing a vulnerability forecast for category 8.....	137
8.19	Graph showing information about harmonised vulnerability category 8	138
8.20	The first three steps for doing a vulnerability forecast in the VF Prototype completed	141
8.21	Step 4 for doing a vulnerability forecast in the VF Prototype.....	142
8.22	Calculations for the fourth μ -value as displayed in figure 8.21	143
8.23	Step 5 and the final vulnerability forecast result for harmonised vulnerability category 8.....	143
8.24	Graph showing information about harmonised vulnerability category 8	144
8.25	The completed VF Prototype main window.....	145
8.26	Comparing a vulnerability forecast with an actual scan – Scan 16	146
8.27	Comparing a vulnerability forecast with an actual scan for a significant time interval increase of 45 days between scans instead of every day	147
A.1	Entering the command to run the VF installation software	157
A.2	Specifying the destination folder for installing the VF Prototype software..	157
A.3	The VF Prototype files being installed by the installation software	158
A.4	Running the register command.....	159
A.5	Successful component registration.....	159
B.1	Project layout of the VF Prototype	161
B.2	The “frmCalculations” form	162

B.3	The “frmGraphics” form	163
B.4	The “frmHelp” form.....	164
B.5	The “frmMain” form.....	170
B.6	The “frmOptions” form	195
B.7	The “frmSaveLoad” form.....	204
B.8	The “frmSelectCats” form.....	213
B.9	The “frmSetup” form	217
B.10	The “frmSetupNames” form	239
B.11	The “frmSWSetup” form.....	247
E.1	Vulnerability history scan data – scan 1	395
E.2	Vulnerability history scan data – scan 2.....	396
E.3	Vulnerability history scan data – scan 3.....	396
E.4	Vulnerability history scan data – scan 4.....	397
E.5	Vulnerability history scan data – scan 5.....	397
E.6	Vulnerability history scan data – scan 6.....	398
E.7	Vulnerability history scan data – scan 7.....	398
E.8	Vulnerability history scan data – scan 8.....	399
E.9	Vulnerability history scan data – scan 9.....	399
E.10	Vulnerability history scan data – scan 10.....	400
E.11	Vulnerability history scan data – scan 11	400
E.12	Vulnerability history scan data – scan 12.....	401
E.13	Vulnerability history scan data – scan 13.....	401
E.14	Vulnerability history scan data – scan 14.....	402
E.15	Vulnerability history scan data – scan 15.....	402
E.16	Vulnerability history scan data – scan 16.....	403
E.17	Scan results over the 16 scans for CyberCop vulnerability category 1	404
E.18	Scan results over the 16 scans for CyberCop vulnerability category 2.....	404
E.19	Scan results over the 16 scans for CyberCop vulnerability category 3.....	405
E.20	Scan results over the 16 scans for CyberCop vulnerability category 4.....	405
E.21	Scan results over the 16 scans for CyberCop vulnerability category 5.....	406
E.22	Scan results over the 16 scans for CyberCop vulnerability category 6.....	406
E.23	Scan results over the 16 scans for CyberCop vulnerability category 7	407
E.24	Scan results over the 16 scans for CyberCop vulnerability category 8.....	407

E.25	Scan results over the 16 scans for CyberCop vulnerability category 9.....	408
E.26	Scan results over the 16 scans for CyberCop vulnerability category 10.....	408
E.27	Scan results over the 16 scans for CyberCop vulnerability category 11.....	409
E.28	Scan results over the 16 scans for CyberCop vulnerability category 12.....	409
E.29	Scan results over the 16 scans for CyberCop vulnerability category 13.....	410
E.30	Scan results over the 16 scans for CyberCop vulnerability category 14.....	410
E.31	Scan results over the 16 scans for CyberCop vulnerability category 15.....	411
E.32	Scan results over the 16 scans for CyberCop vulnerability category 16.....	411
E.33	Scan results over the 16 scans for CyberCop vulnerability category 17.....	412
E.34	Scan results over the 16 scans for CyberCop vulnerability category 18.....	412
E.35	Scan results over the 16 scans for CyberCop vulnerability category 19.....	413
E.36	Scan results over the 16 scans for CyberCop vulnerability category 20.....	413
E.37	Scan results over the 16 scans for CyberCop vulnerability category 21.....	414
E.38	Scan results over the 16 scans for CyberCop vulnerability category 22.....	414
E.39	Scan results over the 16 scans for CyberCop vulnerability category 23.....	415
E.40	Scan results over the 16 scans for CyberCop vulnerability category 24.....	415
E.41	Scan results over the 16 scans for CyberCop vulnerability category 25.....	416
E.42	Scan results over the 16 scans for CyberCop vulnerability category 26.....	416
E.43	Scan results over the 16 scans for CyberCop vulnerability category 27.....	417
E.44	Scan results over the 16 scans for CyberCop vulnerability category 28.....	417
E.45	Scan results over the 16 scans for CyberCop vulnerability category 29.....	418
E.46	Scan results over the 16 scans for CyberCop vulnerability category 30.....	418
E.47	Scan results over the 16 scans for CyberCop vulnerability category 31.....	419



LIST OF TABLES

2.1	Resources covering the information security technologies	15
2.2	The information security technologies	16
4.1	Summary of the harmonised vulnerability categories	49
5.1	State-of-the-art VS products	62
5.2	Harmonised vulnerability categories covered by CyberCop Scanner.....	64
5.3	Harmonised vulnerability categories covered by Cisco Secure Scanner.....	65
5.4	Harmonised vulnerability categories covered by SAINT	67
5.5	Harmonised vulnerability categories covered by ISS	69
5.6	Harmonised vulnerability categories covered by Nessus Security Scanner...	71
5.7	Important network and system information gathering vulnerabilities	74
5.8	Important backdoors, Trojans, and remote controlling vulnerabilities.....	75
5.9	Important misconfiguration vulnerabilities	76
5.10	Important denial-of-service (DoS) and buffer overflow vulnerabilities.....	77
5.11	Important security policy violations vulnerabilities	78
6.1	Problems identified and addressed regarding state-of-the-art VS products....	84
6.2	Summary of the harmonised vulnerability categories	93
7.1	The fuzzy groups formed for harmonised vulnerability category K.....	110
7.2	Mapping table	110
7.3	Distribution of scans and defuzzified ranges	112
7.4	Transforming the defuzzified values using the membership function $\chi(x)$..	113
7.5	Results for the μ 's and χ 's.....	114
7.6	An example FEI calculated for each harmonised vulnerability sorted in order of highest to lowest priority.....	115
8.1	Example of a vulnerability uncovered during a scan	122
8.2	CyberCop vulnerability categories	122

8.3	Problems identified and addressed regarding state-of-the-art VS products	148
A.1	Typefaces employed to indicate special text	156
D.1	The CyberCop Scanner vulnerability database.....	309



CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

The Internet potentially is an invaluable and inexhaustible resource, accessible to each and every person. Almost any conventional publishing media, such as books, journals and magazines, can all be located on the Internet in electronic form these days. The Internet has made life easier in many ways – it has become part of our lives. But, alas, like human nature, it also harbours a dark side, providing – at best – the perfect tool for innocuous pranks such as amateurish attempts at hacking and – at worst – the perfect breeding ground for pernicious cyber-crime schemes and vicious security attacks.

One should accept that there are always pranksters, better known as **hackers** [CRMC 01], who want to steal information for unethical purposes, or simply jeopardise the organisation by making their system resources unavailable. It is for these reasons that a new research field has evolved over the past decade – **information security** [INFO 02].

The application of information security enabled businesses to start conducting business over the Internet. Given enough time and resources, however, any information security application can be cracked. This introduces new and challenging problems manifested in the field of information security. This study was, therefore, primarily motivated by the need for better information security applications, specially those in the realm of network security.

1.2 MOTIVATION FOR THIS STUDY

The research undertaken for this study was motivated by a number of realisations. These realisations are discussed in the sections that follow.

Open environment of the Internet

The Internet is a public network. This means that anyone can possibly intercept messages that travel along the Internet in a bid to spy or steal information. The Internet being a public network, however, does not mean that no private information can be sent across the Internet. There are, in fact, numerous information security services [INFO 02] which are used to implement security measures over the Internet. One such service that is particularly used to keep messages sent over the Internet private is referred to as *confidentiality*. The specific mechanism used to implement confidentiality is referred to as *cryptology* [PHLE 03].

Currently the Internet is working on a specific protocol referred to as the Internet Protocol (IP) version 4 [IPFA 03]. IP version 4 has been employed since the late 1980s. This version of IP, however, has some design flaws [IPVE 03] in that it was initially developed by the American Department of Defence as a private network. Hence, it was not initially designed to incorporate security features and, therefore, security features had to be added onto the application layer of the Internet ISO model [GOLL 99] when the Internet became a public network. Currently a new version of IP – IP version 6 [IPV6 03] – is being developed and will probably replace IP version 4 in the future. IP version 6 is specifically being designed to incorporate security features.

The fact that the Internet is an open environment led to the second realisation.

Rapidly changing environment of the Internet

Probably one of the biggest drawbacks of products, applications and the Internet itself is that they advance so rapidly that information security products struggle to keep up with the pace [SCHU 03]. Naturally, services and applications are being developed and implemented on the Internet – some applications with built-in security features and others without any security features. The reason for some services and applications not having security features built in initially, or having security features built in but not exhaustively, is one of business processes: it is often more important for an organisation to get the business up and running than to wait a while longer and have exhaustive security features implemented. Whether security has been

implemented in the services and applications or not, only after installation and functioning of such services and applications are security holes – referred to as **vulnerabilities** – found, most of the time, by hackers [SCHN 00].

If hackers find vulnerabilities first, they will exploit them, which could result in the theft, loss or corruption of data. If security experts find vulnerabilities first, they will create additional code, referred to as a security patch, to rectify the vulnerability. Software vulnerabilities are found and exploited like clockwork by hackers. In response, software patches become available too, but often organisations are the victims of hacker attacks because of this rapidly changing of the Internet environment.

The rapidly changing environment of the Internet led to the next realisation.

A legion of security products available

Because of the rapidly changing environment of the Internet, the number of security products available on the software market today is legion. For almost any security risk that is currently known, security applications have already been developed for it. Some security products implement information security technologies that have been known to people for ages, for example the Caesar cipher [PHLE 03] named after Julius Caesar. Most recent security products, however, implement information security technologies that have been known to people for only the past decade as a result of the advent of the Internet, for example intrusion detection and vulnerability scanner information security technologies.

Although the implementation of older information security technologies has been perfected to a reasonable extent, current information security products still need to be perfected. These current information security products fall short in many ways, because they are still very new. Considerable room for perfecting these information security products is therefore possible. Too many false alarms, responses that are not prompt, too much redundant work and the huge reports generated [SCHN 00] are just some of the areas of information security products that sorely need attention.

Although the information security products need attention in the areas described above, the researcher also realised that there are many implementations by different vendors of the same information security technology, as the following realisation confirms.

Disparity in similar security products

There are often different vendors that create security products for the same security service or application. For example, in the application of cryptology, the following are a few examples of cryptology products and standards created by different vendors, which, in essence, perform the same functions – that of encryption and decryption: Privacy Master [WEBR 03], Pretty Good Privacy (PGP) [PGPI 03], Data Encryption Standard (DES) [WEBD 03] and Advanced Encryption Standard (AES) [WEBA 03].

The products mentioned here are all similar in terms of their application, but they are disparate in the way they are implemented. For example, Privacy Master uses symmetric key encryption [WEBS 03], while PGP uses asymmetric key encryption [WEBP 03]. Even more finely disparate, DES uses a key length of 56 bits, while AES uses a key length of 128 bits. This disparity in security products often harbours confusion amongst the users and potential users of the different security products in terms of which product is better to use, or which product is the right one to use according to the needs of the user or organisation.

The disparity in similar security products is one realisation. However, some of these security products require a colossal effort from an administrator's point of view to manage, which led to the next realisation.

Huge administrative burden regarding vulnerability scanner products

The nature of security products often involves a huge administrative burden. For example, an information security product known as a **vulnerability scanner** can reveal vulnerabilities by scanning for them on computers connected to a network. Depending on the number of computers that are scanned during a vulnerability scan,

often thousands of vulnerabilities are detected, which results in the generation of a colossal report consisting of hundreds or even thousands of pages. It is left to the responsible administrator to analyse such a report in a bid to rectify the vulnerabilities found. The administrative burden to do this, therefore, is huge. The result is often that, because of this huge administrative load, security cannot be applied as anticipated due to a shortage of human resources and, therefore, security is often neglected.

The above realisations strengthened the researcher's resolve to develop a model specifically aimed at vulnerability scanning, which would facilitate the job of an administrator and render current vulnerability scanners more effective.

1.3 PROBLEM STATEMENT

This research recognises the importance of information security and specifically that of current information security technologies. It is aimed principally at making a contribution to enhancing risk management by forecasting the extent to which vulnerabilities will occur in the future. A model for vulnerability forecasting is, therefore, proposed that follows a fresh approach to vulnerability assessment.

The problem area can be addressed by considering the following research questions:

What is the state of current proactive and reactive information security technologies?

State-of-the-art information security technologies each implement one or more of the five information security services: authentication, confidentiality, integrity, availability and non-repudiation. In doing so, information is secured by the information security technologies either on a proactive basis by securing information before it can be compromised, or on a reactive basis by securing information as soon as an attempt is made to compromise the information.

The investigation of this research question, therefore, will involve a detailed analysis of state-of-the-art information security technologies with the aim of categorising them into proactive and reactive technologies. Furthermore, it should be investigated

whether proactive or reactive information security technologies will be used as the platform to conduct this research project. In order to accomplish this, a detailed study will be conducted on one proactive and one reactive information security technology.

What can be done to improve the vulnerability scanning process?

State-of-the-art vulnerability scanners scan for vulnerabilities and report on their findings after a scan is complete. It is difficult and time-consuming, however, to effectively attend to the vulnerabilities reported after such a vulnerability scan, because such reports are often very long and left entirely up to human resources to rectify, leaving them with an immense administrative burden.

The investigation of this research question would therefore involve finding techniques in order to ease the administrative burden on human resources.

How can the impact of current vulnerability scanners on system resources be minimised?

Current vulnerability scanners detect vulnerabilities by scanning for signatures of known vulnerabilities and attack patterns. The modus operandi of a vulnerability scanner is often to simulate attacks that would attack system resources of a computer or network of computers to test whether the system resources have been secured sufficiently and, if not, how these computers would react to genuine attacks. This way of detecting vulnerabilities often has the same effect of genuine attacks and, therefore, could deny the services of system resources significantly or entirely.

The impact that current vulnerability scanners have on system resources, therefore, may result in the inability of normal business processes to continue due to the interruption of system resources by a vulnerability scan. The investigation of this research question will involve alternative scanning strategies of current vulnerability scanners.

How can the disparity be addressed in the kinds of vulnerabilities that different vulnerability scanner products can detect?

There is a disparity in current vulnerability scanners in the way that they detect vulnerabilities. An example of a specific area of disparity is vulnerability scanner **X** being able to scan for specific vulnerabilities, whereas vulnerability scanner **Y** scanning for different kinds of vulnerabilities.

The investigation of this research question will therefore involve finding techniques in order to standardise the kinds of known vulnerabilities so that, ultimately, it is possible to know which subset of standardised vulnerabilities a specific vulnerability scanner can detect from a potentially exhaustive set of standardised vulnerabilities.

How should vulnerability scanner products provide more intelligent results so that they will aid risk management?

Current vulnerability scanners are lacking in the sense that they are not able to supply management of an organisation with sufficient results that would enable them to engage in risk management of their information more effectively. An example of “more intelligent results” is that a vulnerability scanner is able to predict the vulnerability trends the organisation can expect in the near future.

This research question, in a way, summarises the research questions stated above, since solving those questions would enable the researcher to gather sufficient knowledge about proactive information security technologies so that more intelligent techniques can be applied to solve this research question.

Although risk management is a component that is included in the model proposed in this thesis, it is not the aim of this thesis to present a full-blown discussion on risk management. However, the outcome of this thesis would assist human resources when engaging in risk management and, at the heart of being able to provide this outcome, techniques which employ fuzzy logic are used [BOBO 95, SMIT 00, YAZA 92, ZADE 65].

1.4 TERMINOLOGY USED IN THIS THESIS

It is important to correctly interpret the terminology used in this thesis to avoid misunderstanding. Detailed terminology will be defined when encountered throughout the thesis. However, to ensure that the main terms are well defined, the researcher will now provide a brief delineation of what is meant by the terms **information security**, **intrusion detection**, **vulnerability scanning**, and **risk management**.

1.4.1 Information security

Information, like other important business assets, is an asset that has value to an organisation and consequently needs to be protected [BSIB 03]. Assets, in this context, may include knowledge, facts, data or capabilities. Capabilities can refer to an event that involves the handling of information, for example sending a message. **Information security** can be defined as measures adopted to prevent the unauthorised use, misuse, modification or denial of the use of assets [MAIW 03]. Furthermore, the objective of information security is not to protect assets, but it is the name given to the preventative measures that can be taken to safeguard assets [IFAC 98].

The measures that information security employs in order to prevent the unauthorised use, misuse, modification or denial of the use of assets are known as the five **information security services** as described below [GOLL 99, ISOR 89]:

- **Authentication** is concerned with a process or method to identify and prove the identity of a party who attempts to send a message or access data.
- **Confidentiality** is concerned with the protection of information against disclosure to an unauthorised party.
- **Integrity** is concerned with the protection of information against being changed by an unauthorised party.
- **Availability** is concerned with information being made available to authorised parties when requested.
- **Non-repudiation** is concerned with providing proof of the origin such that the sender cannot deny sending a specific message, and the recipient cannot deny receiving that message.

Any security product that is developed implements the five information security services to a certain extent by means of technologies, be they hardware or software technologies. **Technology** refers to “the application of science, especially to industrial or commercial objectives” [LEXI 02]. **Information security technology** refers to the application of all possible state-of-the-art security technologies to all possible information [INFO 02].

This thesis ascribes to two specific information security technologies, namely intrusion detection and vulnerability scanning. The following two sections define these two terms.

1.4.2 Intrusion detection

An **intrusion** is any set of actions that attempt to compromise the integrity, confidentiality or availability of a resource. **Intrusion detection** is the process of monitoring the events that occur in a computer system or network and analysing them for signs of intrusions [BACE 00]. Intrusion detection is considered a *reactive* information security technology, because only after the event of an intrusion occurred will there be a reaction to the event. An **intrusion detection system (IDS)** is a software product or hardware technology that automates the monitoring process [KIDO 01].

The counter technology for intrusion detection is known as **vulnerability scanning**, which is defined in the next section.

1.4.3 Vulnerability scanning

A **vulnerability** is a state of being exposed to attack, injury, ridicule or litigation [LEXI 03]. In the context of this thesis, a vulnerability is a known weakness in a computer system that is exposed to attack, which can be exploited by a hacker. To **scan** means “to examine closely” [LEXI 03] and a **vulnerability scanner (VS)** is an automated scanning program that closely examines or scans a computer or a network of computers to *proactively* detect known vulnerabilities [SCHN 00]. **Vulnerability scanning**, therefore, is an information security technology implemented by a VS information security product. Vulnerability scanning is often alternatively referred to

as **vulnerability assessment**, but the terms **vulnerability scanning** and **vulnerability scanner** are preferred and will be used throughout this thesis.

The *proactive* concept refers to those information security technologies that attempt to deal with information security issues *before* any attempt can be made by an attacker to break into or harm a system. Proactive information security technologies may assist with risk management, because risk management can also be considered as a proactive process where risks are identified before they can occur. Risk management is defined in the section that follows.

1.4.4 Risk management

The term “**risk**” means “the possibility of suffering harm or loss” [LEXI 03]. **Risk management**, in the context of this thesis, therefore, is the process that allows one to identify threats and risks and then eliminate those that can be eliminated and minimise the rest [BACE 00]. The threats and risks refer to vulnerabilities in terms of this thesis.

The rest of the thesis is laid out as discussed in the section that follows.

1.5 LAYOUT OF THESIS

This thesis consists of nine chapters. The current chapter, **chapter 1**, provides an introduction to the research problem. In **chapter 2**, the reader is provided with a taxonomy for information security technologies. Special reference is made in this chapter to the two main aspects of this taxonomy, namely *proactive* and *reactive* information security technologies.

In the next chapter, **chapter 3** two specific information security technologies are discussed – one reactive and one proactive information security technology. The reactive information security technology discussed is *intrusion detection*, while *vulnerability scanning* is the proactive information security technology discussed. For each of these technologies, an overview is provided, followed by an architectural description of the technology itself as well as alternative architectures. After that, the

problems of the particular information security technology are discussed, followed by some examples of commercially available information security products for the particular information security technology.

One of the major problems, as identified in chapter 1 and discussed in chapter 3, is tackled in **chapter 4** – standardising vulnerability categories so that harmonised vulnerability categories are formed. The chapter describes the method used to compile such harmonised vulnerability categories, and then discusses each of the categories in detail.

In order to see how the harmonised vulnerability categories can be applied, **chapter 5** provides an overview of current VS products and then discusses the impact of the harmonised vulnerability categories on the VS products. In addition, the researcher describes how each VS product was practically experienced and provides comments on the vulnerability database of each. Thereafter, specific differences in these VS products are pointed out using the harmonised vulnerability categories.

Chapter 6 continues to address the rest of the problems as stated in the problem statement by introducing the concept of vulnerability forecasting. In this chapter, a conceptual model for doing vulnerability forecasting is proposed. The design of the model is discussed in detail while specific reference is made to the design of the database used for vulnerability forecasting.

One of the components that forms the heart of the vulnerability forecasting model, the vulnerability forecast engine, is discussed in detail in **chapter 7**. This chapter explains the input that the vulnerability forecast engine receives, and how the input is transformed using five sophisticated steps and fuzzy logic techniques in order to produce as output the vulnerability forecast.

The thesis culminates in **chapter 8** when the model for vulnerability forecasting is tested using a prototype for vulnerability forecasting. This chapter first explains the extent to which the prototype was developed and implemented according to the vulnerability forecasting model. It then explains how the prototype can be installed

and executed. Furthermore, the chapter demonstrates the operation of the prototype in detail and reports on the findings of the prototype.

The thesis summarises the research undertaken in **chapter 9** and explains the extent to which the research problem has been solved. The thesis concludes with a reflection on possible areas for future research.

Finally, appendices are given, followed by a bibliography of resources consulted for this research.



CHAPTER 2

A TAXONOMY FOR INFORMATION SECURITY TECHNOLOGIES

2.1 INTRODUCTION

As the Internet took the world by storm in the mid-1990s, so did security problems. Unfortunately, hackers developed their own software which enabled them, for example, to sniff a password being sent over the Internet. In another example, a hacker might send malicious data over the Internet so that servers connected to the Internet will not be able to handle such malicious data and the servers will simply fail.

Fortunately, intensive research in computer and Internet security has proved to deliver countermeasure technologies, better known as information security technologies, over the past decade for the majority of these and other security problems. This chapter provides a taxonomy of information security technologies available today.

The sections that follow will give a taxonomy of the information security technologies available today, after which each technology is briefly explained.

2.2 A TAXONOMY FOR INFORMATION SECURITY TECHNOLOGIES

What is **information security technology**? *Information security* involves the protection of information [MASI 02] and minimises the risk of exposing information to unauthorised parties [KIDO 01]. According to Dictionary.com, *technology* is “the application of science, especially to industrial or commercial objectives” [LEXI 02]. *Information security technology* thus refers to the application of all possible state-of-the-art security technologies to all possible information [INFO 02].

Figure 2.1 shows a taxonomy of information security technologies. A taxonomy is the classification of objects in an ordered list or hierarchy of terms that indicates natural relationships [COSL 02, LEXI 02]. This taxonomy is based primarily on two characteristics:

1. The specific point in time, namely proactive or reactive, when the technology interacts with data.
2. Whether the technology interacts at network, host, or application level.

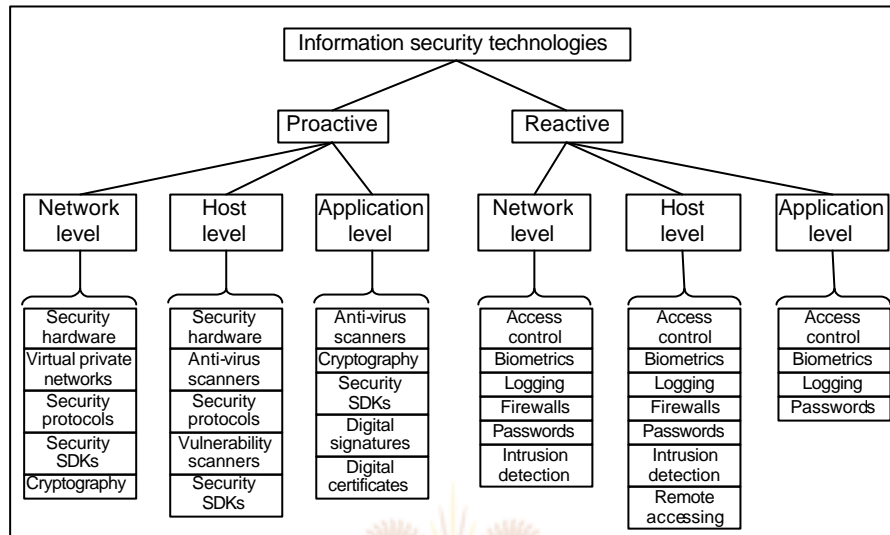


Figure 2.1: A taxonomy of information security technologies

Proactive means that preventative measures have been taken by the specific information security technology in a bid to secure data or resources before a security breach can occur. *Reactive* means that curing measures are being taken by the specific information security technology in a bid to secure data or resources as soon as a security breach is detected. Both proactive and reactive information security technologies can apply to *network*, *host*, or *application* level. Information security technologies at *network level* attempt to secure data or resources being transmitted over a system of computers interconnected by telephone wires or other means in order to share information. Information security technologies at *host level* attempt to secure data or resources that reside on a single computer. Information security technologies at *application level* attempt to secure data or resources that specifically relate to a single computer program on a host.

A comprehensive literature study was conducted to identify the state-of-the-art information security technologies available. This is indicated in table 2.1. A distinction was made between journals and books. The objective was to firstly identify which technologies are addressed by the different resources and secondly the degree to which these technologies are addressed. Whenever a specific information security technology was addressed by a specific resource, it was taken into account. A tick mark shown in table 2.1 appears only when the specific technology is addressed comprehensively by a specific resource.

Table 2.1: Resources covering the information security technologies

Resource	Access control	Biometrics	Remote access	Passwords	Cryptography	Digital signatures	Digital certificates	Firewalls	Virtual private networks	Intrusion detection systems	Vulnerability scanners	Anti-virus scanners	Security SDKs	Logging	Security protocols	Security hardware
Journals																
Computers & Security [COMP 02]	✓						✓						✓	✓	✓	✓
Computer Fraud & Security [FRAU 02]	✓			✓	✓	✓	✓			✓	✓	✓	✓	✓	✓	✓
Network Security [NETW 02]		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Books																
Internet & TCP/IP Network Security [PAGU 96]			✓					✓		✓						✓
Secure Communicating Systems [HUTH 01]					✓											
Computer Security Policies [WACA 98]						✓	✓	✓							✓	
Windows 2000 Security [MCLE 00]	✓		✓		✓	✓	✓	✓					✓	✓	✓	✓
Hackers Beware [COLE 02]				✓							✓				✓	
Computer Security [CARR 96]	✓			✓	✓							✓			✓	
Hacking Exposed [MCSK 02]			✓	✓				✓		✓	✓	✓	✓	✓		
Intrusion Detection [BACE 00]										✓	✓				✓	✓
Network Intrusion Detection [NONM 01]										✓	✓					✓
Access Denied [CRMC 01]	✓	✓		✓	✓	✓		✓			✓					
Internet & Intranet Security [OPPL 98]						✓		✓			✓					✓
Secrets & Lies [SCHN 00]		✓			✓						✓	✓	✓			✓
Security Architecture [KIDO 01]	✓					✓	✓	✓	✓	✓	✓	✓	✓	✓		✓
Security in Computing [PHLE 03]	✓				✓	✓	✓	✓				✓				✓
Computer Security [GOLL 99]	✓			✓		✓						✓	✓			
Information Security Architecture [TUDO 00]	✓	✓	✓	✓				✓	✓			✓				
Web Security [STEI 98]	✓		✓	✓		✓	✓	✓				✓	✓	✓		
Web Security [TIWA 99]			✓			✓	✓					✓	✓	✓		

The information security technologies are listed in table 2.2 and a brief description of each of these technologies is given in the sections that follow.

2.2.1 Proactive information security technologies

Proactive information security technologies take preventative measures by securing data or resources *before* a security breach can occur. The sections that follow sets out to describe each proactive information security technology listed in table 2.2.

Table 2.2: The information security technologies

2.1 Proactive information security technologies
2.1.1 <i>Cryptography</i>
2.1.2 <i>Digital signatures</i>
2.1.3 <i>Digital certificates</i>
2.1.4 <i>Virtual private networks</i>
2.1.5 <i>Vulnerability scanners</i>
2.1.6 <i>Anti-virus scanners</i>
2.1.7 <i>Security protocols</i>
2.1.8 <i>Security hardware</i>
2.1.9 <i>Security SDKs</i>
2.2 Reactive information security technologies
2.2.1 <i>Firewalls</i>
2.2.2 <i>Access control</i>
2.2.3 <i>Passwords</i>
2.2.4 <i>Biometrics</i>
2.2.5 <i>Intrusion detection systems</i>
2.2.6 <i>Logging</i>
2.2.7 <i>Remote accessing</i>

2.2.1.1 Cryptography

Cryptography, in simple terms, means “hidden writing”. It is the science of protecting data confidentiality and integrity [MCSK 02]. *Encryption* is the process of transforming or scrambling a cleartext message so that it becomes a ciphertext message. Synonyms for encryption are *encode* and *encipher*. The reverse process of encryption is called *decryption*, which is the process of rearranging the ciphertext so that a ciphertext message is transformed into a cleartext message. Synonyms for decryption are *decode* and *decipher*.

Cryptography is a *proactive* information security technology because it safeguards data before a potential threat can materialise by encrypting the data. This is done to prevent an intruder from tapping a network wire and sniffing sensitive information from the network. Furthermore, cryptography is performed at various levels as indicated by the taxonomy:

- At application level: A specific application performs the encryption process before an intruder is able to intercept sensitive data.
- At network level: Hardware rather than software encryption can take place where hardware encryption modules can be placed at network level.

2.2.1.2 Digital signatures

A digital signature can be thought of as the equivalent of a handwritten signature with the same goal: associating a mark that is unique to an individual with a body of text [PHLE 03]. In the same way as a handwritten signature, a digital signature must not be forgeable, in other words only the legitimate sender of a message should be able to create the digital signature [KIDO 01]. Digital signatures are created using cryptographic algorithms.

A digital signature is a *proactive* information security technology because the digital signature is created before any dispute can arise that a specific sender of a message is not really the intended sender. Creating a digital signature thus indicates beforehand that a specific sender of a message is the sole creator of that message. Furthermore, a digital signature is created at the following level as indicated by the taxonomy:

- At application level: The digital signature is created by a specific application before it is sent off to a specific receiver.

2.2.1.3 Digital certificates

Digital certificates attempt to solve the problem of *trust* on the Internet. They are issued by *trusted third parties*, also referred to as *certificate authorities* (CAs) [TIWA 99]. CAs are commercial enterprises that *vouch* for the identities of people or organisations on the Web [STEI 98]. A network of trust is thus established amongst Web users. In simple terms the concept of “trust” or “vouching for” can be stated as “someone I trust – the CA – trusts this other person, so I will trust him as well” [PHLE 03].

A digital certificate is a *proactive* information security technology because the certificate is used to distribute the public key of a communicating party to another communicating party. In this way trust is also established before any communication

between parties takes place. Furthermore, a digital certificate is implemented at the following level as indicated by the taxonomy:

- At application level: A specific application, for example a Web browser, verifies that it can trust a specific party before communication commences.

2.2.1.4 Virtual private networks

Virtual private network (VPN) technology encrypts network traffic and therefore the technology is closely related to cryptography. A VPN allows an organisation with multiple sites to connect these sites over a public network, i.e. the Internet, with the advantages that all data packets that travel between the sites are encrypted and secure [COME 99]. In addition, the packets are restricted by the VPN technology to only travel between the organisation's sites. The difference in functionality between normal encryption and VPNs, however, is that the data is encrypted only when it is transmitted over a public network – the data that travels between the originating host and the VPN host is not encrypted. In addition, data will only be encrypted by the VPN if it originates from an authenticated host.

A VPN is a *proactive* information security technology because it safeguards data before it is transmitted over a public network by encrypting it so that only legitimate persons are able to read the information. Furthermore, VPNs work at the following level as indicated by the taxonomy:

- At network level: The encryption process is done between two VPN hosts sitting on the points-of-entry in a network before the encrypted data is sent over a network.

2.2.1.5 Vulnerability scanners

Vulnerability scanners (VSs) use signatures for the vulnerabilities they can identify. Therefore, a VS is an information security technology which is but a special case of intrusion detection [BACE 00]. Vulnerability scanning is also referred to as *interval-based* scanning, because hosts on a network are scanned at certain intervals and not continuously. When a VS has completed a scan and sampled the data into a report, it is referred to as a *snapshot*.

A VS is a *proactive* information security technology because it attempts to identify vulnerabilities before they can be exploited by intruders or malicious applications. Furthermore, VSs work at the following level as indicated by the taxonomy:

- At host level: A VS scans for vulnerabilities across an entire host in a bid to identify vulnerabilities in all the software applications and the hardware of the specific host.

2.2.1.6 Anti-virus scanners

Computer viruses have caused havoc on the Internet over the past decade. A computer virus is a piece of malicious software which has the ability to reproduce itself across the Internet, once activated [MCSK 02]. Therefore anti-virus scanners have been developed to counteract computer viruses.

Anti-virus scanners attempt to scan for viruses and functions before they can cause havoc, much in the same way as VSs in that they also “know” what a specific virus’s signature looks like. Anti-virus software is therefore also a *proactive* information security technology. Furthermore, anti-virus scanning is performed at various levels as indicated by the taxonomy:

- At application level: A specific application scans for known virus signatures in an effort to detect them before they can cause havoc. Viruses at application level tend to be Trojan horses, because they are hidden in an application and only activates once that application is executed; they do not reproduce themselves.
- At host level: Viruses that have the ability to reproduce themselves by using e-mail applications, for example, can cause malicious activity almost anywhere on a host. Such viruses need to be scanned for across the entire host before they can start their malicious activity.

2.2.1.7 Security protocols

There are different protocols, for example Internet Protocol Security (IPSec) and Kerberos, that can be classified as information security technologies. These protocols are technologies that use a standard procedure for regulating data transmission between computers or applications to safeguard sensitive information before such information can be intercepted by intruders.

Security protocols are *proactive* information security technologies because they attempt to safeguard sensitive information using a specific security protocol before such information can be intercepted by intruders. Furthermore, security protocols work at various levels as indicated by the taxonomy:

- At application level: A security protocol, for example Kerberos, is a mutual authentication protocol which handles authentication at application level.
- At network level: A security protocol also relies on a network infrastructure to perform its security task, whether it is to encrypt data or simply to encapsulate a network packet in an effort to hide the packet's identity for security purposes.

2.2.1.8 Security hardware

Security hardware refers to physical hardware devices used to perform security tasks, for example hardware encryption modules or hardware routers.

Security hardware is a *proactive* information security technology because it safeguards data before a potential threat can materialise by, for example, encrypting data. This is done to prevent an intruder from changing or modifying the hardware device, since security hardware consists of physical devices that are tamper-proof. Furthermore, security hardware is implemented at various levels as indicated by the taxonomy:

- At host level: A hardware device can be attached to a specific host to perform its security function, for example a hardware key could be inserted into a specific port of a host to authenticate a specific user before the user is able to log on to the host.
- At network level: Hardware encryption modules can be placed on the network, which provides a tamper-proof solution, and can be physically secured.

2.2.1.9 Security SDKs

Security software development kits (SDKs) are programming tools used to create security programs. The Java security manager and Microsoft .NET SDKs are examples of software that can be used to build security applications such as Web-based authentication programs.

Security SDKs are *proactive* information security technologies because they are used to develop various software security applications that safeguard data before a potential threat can materialise. Furthermore, security SDKs are used to develop security software at various levels as indicated by the taxonomy:

- At application level: A specific software application can be developed to safeguard data by encrypting data on disk, for example.
- At host level: A specific software application can be developed to authenticate a user or a process to a host.
- At network level: A specific software application can be developed to safeguard data by encrypting it before sending it over a network, for example.

2.2.2 Reactive information security technologies

Reactive information security technologies take curing measures by securing data or resources as soon as a security breach is detected or after such a security breach has occurred. The sections that follow sets out to describe each reactive information security technology listed in table 2.2.

2.2.2.1 Firewalls

An Internet firewall is a software tool installed on a specially configured computer that serves as a blockade, filter, or bottleneck between an organisation's internal or trusted network and the untrusted network or Internet [TIWA 99]. The purpose of a firewall is to prevent unauthorised communications into or out of the organisation's internal network or host [OPPL 98]. Firewalls are considered as the first line of defence in keeping intruders out [PAGU 96]. Personal firewalls are new to the security arena. Unlike traditional firewalls, personal firewalls are installed on a normal workstation and attempt to only protect that specific workstation from the rest of the hosts on the network or the Internet.

Firewalls are *reactive* information security technologies because they are used to act against specific security incidents as soon as they occur. Furthermore, firewalls are implemented at various levels as indicated by the taxonomy:

- At host level: A personal firewall can be installed on a host that attempts to block or allow certain data flow to and from that specific host only.

- At network level: A network firewall can be installed on a host that is acting as the gateway to a private network. A network firewall attempts to block or allow certain data flow to and from all the hosts situated behind the network firewall.

2.2.2.2 Access control

The goal of access control is to ensure that a subject has sufficient rights to perform certain actions on a system [KIDO 01]. A subject may be a user, a group of users, a service, or an application. Subjects have different levels of access to certain objects in a system. An object may be a file, a directory, a printer, or a process.

Access control is a *reactive* information security technology because it is used to allow or deny access to a system as soon as an access request is made. Furthermore, access control is implemented at various levels as indicated by the taxonomy:

- At application level: Access is allowed or denied to subjects on access requests to specific objects using access control lists in an application.
- At host level: Access is allowed or denied to a host when a user attempts to log on to the host.
- At network level: Access is allowed or denied to the network when a user attempts to log on to the network through a host or process.

2.2.2.3 Passwords

A password is a secret word, phrase, or sequence of characters that one must input to gain admittance or access to information such as a file, application, or computer system [LEXI 02]. Passwords, however, should be considered as a technology on its own since the literature, as presented in table 2.1, does so.

Passwords are *reactive* information security technologies because they are used to allow or deny access to a system as soon as a person or a process wants to log on to an application, host, or network. Furthermore, passwords are implemented at various levels as indicated by the taxonomy:

- At application level: A person or process is allowed or denied access to a specific application, depending on whether the person or process provides the correct password.

- At host level: A person or process is allowed or denied access to a specific host, depending on whether the person or process provides the correct password.
- At network level: A person or process is allowed or denied access to a network, depending on whether the person or process provides the correct password.

2.2.2.4 Biometrics

Biometrics uses the geometry of a specific part of a human body to authenticate a person. There are many different implementations of biometrics, for example hand, fingerprint, retina and voice recognition biometrics.

Biometrics is a *reactive* information security technology because it is used to allow or deny access to a system as soon as a person wants to log on to an application, host, or network using the geometry of a part of his/her human body. Furthermore, biometrics is implemented at various levels as indicated by the taxonomy:

- At application level: A person is allowed or denied access to a specific **application**, depending on whether the person provides his/her own biometric characteristic. For example, a user might be requested to place a finger on a fingerprint reader in order to open a top secret file.
- At host level: A person is allowed or denied access to a specific **host**, depending on whether the person provides his/her own biometric characteristic. For example, a user might be requested to place a finger on a fingerprint reader in order to log onto a workstation.
- At network level: A person is allowed or denied access to a **network**, depending on whether the person provides his/her own biometric characteristic. For example, a user might be requested to place a finger on a fingerprint reader in order to access other hosts or resources across a network domain.

2.2.2.5 Intrusion detection systems

An *intrusion detection system* (IDS) is a software or hardware technology that, once activated, constantly monitors a computer system for intrusions [BACE 00, KIDO 01].

IDSs are *reactive* information security technologies because they are used to monitor hosts on a network and to act on an intrusion as soon as it occurs. Furthermore, IDSs are implemented at various levels as indicated by the taxonomy:

- At host level: An IDS monitors a specific host to detect intrusions on that specific host. It runs on an individual host and continually reviews the host's audit log, looking for possible indications of an intrusion [COLE 02].
- At network level: An IDS node can be placed in a network which attempts to detect and react on intrusions caused by multiple hosts, for example a distributed denial-of-service attack.

2.2.2.6 Logging

Logging is an information security technology that attempts to gather information on certain events that take place. The goal of logging is to supply audit trails which can be traced after a security incident has taken place.

Logging is a *reactive* information security technology because it is used to trace security incidents after they have taken place. Furthermore, logging is implemented at various levels as indicated by the taxonomy:

- At application level: A specific software application monitors other software applications and records the events caused by those software applications.
- At host level: A specific software application monitors the processes that are run by the operating system and records the events caused by those processes.
- At network level: A specific hardware or software application can monitor network traffic as it moves past the network monitor at a specific point in a network.

2.2.2.7 Remote accessing

Remote accessing is an information security technology that allows people or processes to access remote services. However, access to remote services is not always controlled because it is possible to access a remote service anonymously. In this case, accessing remote services anonymously poses a threat. For example, some systems may be wrongly configured to allow anonymous connections by default, when anonymous connections should not actually be allowed according to an organisation's security policy.

Remote accessing is a *reactive* information security technology because it enables a user or process to connect to a remote service according to their access privileges. Furthermore, remote accessing is implemented at the following level as indicated by the taxonomy:

- At host level: A specific host runs a service that enables a remote user or process to connect to it for reasons such as doing remote administration on that host, or legitimately accessing resources on the host.

2.3 CONCLUSION

The taxonomy for information security technologies discussed in this chapter provides an overview of the state-of-the-art information security technologies. It is important for an organisation to know which information security technologies are available.

Furthermore, having such a taxonomy of information security technologies will also stimulate new research. For example, intrusion detection systems are not yet intelligent enough – a human still needs to interact too much in setting up and maintaining intrusion detection systems. In another example, vulnerability scanners take up too many resources and too much time to be effective enough since regular scans need to be conducted for such a technology to be effective.

New initiatives might also be researched, such as combining various information security technologies to form more intelligent ones. For example, it might be possible in the near future to combine firewalls, intrusion detection systems, and anti-virus scanner technologies to form a robust information security technology.

The next chapter will discuss two specific information security technologies in more detail. These two technologies are **intrusion detection** and **vulnerability scanning**.



CHAPTER 3

STATE-OF-THE-ART INTRUSION DETECTION AND VULNERABILITY SCANNING

3.1 INTRODUCTION

These days, there are so many reports of security incidents, for example a hacker that has compromised millions of credit card numbers [HILL 02], or yet another lethal computer virus that has caused the loss of extraordinary amounts of money [PALM 01]. This indicates that computer security is without a doubt a major problem. There are many reasons for this, but, in general, applying computer security in an organisation in the twenty-first century has become a much more difficult task than it was perhaps a decade ago. This is because the Internet expanded much faster than anyone anticipated. The Internet was not initially designed to act as a carrier of public as well as private information and therefore security is a feature that was added only later.

The question is: how secure is the information that resides on a single computer or that travels over a public network? There are many ways in which information can be secured by using information security technologies [VEE1 03], and these were discussed in the previous chapter.

This chapter will address reactive and proactive security measures by using two specific information security technologies: **intrusion detection** as a *reactive* information security technology, and **vulnerability scanning** as a *proactive* information security technology. Although intrusion detection and vulnerability scanning are seen as two different security technologies, there are also similarities between them. The chapter concludes with final remarks on intrusion detection and vulnerability scanning.

3.2 INTRUSION DETECTION

3.2.1 What is intrusion detection?

Intrusion detection is the process of monitoring the events that occur in a computer system or network and analysing them for signs of intrusions [BACE 00]. An *intrusion* is any set of actions that attempt to compromise the integrity, confidentiality, or availability of a resource. An *intrusion detection system* (IDS) is a software or hardware technology that automates this monitoring and analysis process [KIDO 01, GENG 02].

IDSs are *reactive* information security technologies because they attempt to detect an intrusion as soon as it occurs or after it has occurred. Therefore IDSs are sometimes also referred to as *monitors* [BACE 00].

Other systems that are analogous to IDSs are burglar alarms and video surveillance systems. Both these systems and IDSs have one thing in common: attempting to trigger some sort of alarm when an intruder crosses a prohibited boundary.

There are several architectures that are used to build different IDSs. Architecture in this context refers to the overall design, construction, and orderly arrangement of components – specifically of an IDS [LEXI 03]. The reason for the different architectures being employed is a result of the evolving intrusion detection needs over the past years. These architectures are discussed in the next section.

3.2.2 The architecture of IDSs

There are some aspects that play an important role in the architecture of IDSs. These aspects include the following:

- The **location** of the IDS in a network.
- The **data source** that serves as input to the IDS.
- The **analysis engine** that forms the heart of the IDS.
- An **intrusion template database** that contains known templates of intrusions.
- The way in which IDSs **report** their findings.

The above aspects are discussed in detail throughout this section. The data source, analysis engine, database, and report aspects of IDSs also form part of the main components of an IDS.

The typical **location** of an IDS is shown in figure 3.1. IDSs can detect intrusions that occur from a remote or outside network, as well as from an inside, or protected, network.

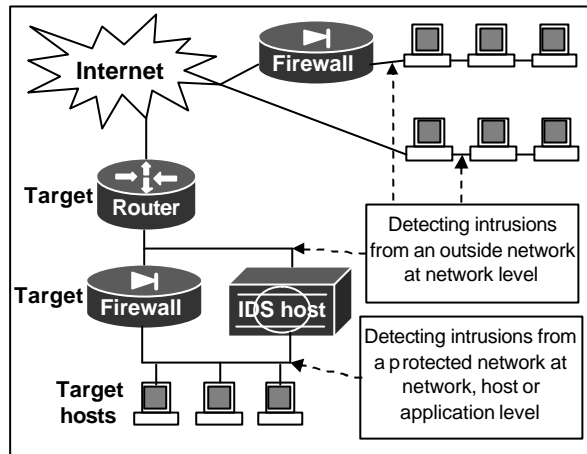


Figure 3.1: The typical location of an IDS in a network

A very important aspect of IDSs is that they require a **data source**, such as applications, hosts, and networks, to collect logged data or network traffic which will be interpreted by the IDS in a bid to detect intrusions. An IDS can monitor for such data at different sources at different levels, as depicted by figure 3.1. However, the data being captured during monitoring is collected by a separate module, referred to as the *IDS host*. The different levels of data sources that an IDS may monitor are referred to as the *targets*. The following are the different targets that an IDS can monitor [COLE 02]:

- *Network-based targets*: The target here is an internal or external network where the IDS sniffs all network traffic crossing over a specified section of a network. While looking at the packets that it sniffs, an IDS looks for signatures that indicate possible intrusions.

- *Host-based targets*: The target here is an individual host. The IDS continually reviews the host's audit log, typically at operating system level, looking for possible indications of an intrusion. Host-based IDSs are operating system-specific.
- *Application-based targets*: The target here is one or more specific applications that are running on a target host. The IDS continually reviews an audit log for the specific application, looking for possible indications of an intrusion.
- *Target-based targets*: The target here is somewhat different from the previous targets in the sense that target-based IDSs generate their own data. This is done, for example, by using cryptographic hash functions to detect alterations to system objects and then, by comparing the alterations to a predefined policy, the IDS can possibly detect an intrusion.

The **analysis engine** is used by the IDS to process the source data. The analysis engine takes the information gathered from the data source and analyses it for signs of intrusion. The modus operandi of the analysis process is to match each piece of data from the data source with a specific template stored in an **intrusion template database**. This database contains different templates of known intrusion techniques. An intrusion is therefore detected as soon as a piece of the source data matches a template intrusion in the intrusion template database. At the same time that an intrusion is detected, it is logged in the form of a detailed **IDS report** of the possible intrusions detected and some IDSs additionally sound an alarm so that a person can interact and deal with the intrusion.

Most IDS approaches include two distinctive architectures: *pattern matching* and *anomaly detection* [DENN 87, COLE 02, ASTI 99]. Both these distinctive architectures, however, contain an *analysis engine* component. The analysis engine forms the heart of any IDS and it is this component that is of particular importance in this research. These two distinctive architectures are discussed in the next two sections.

3.2.2.1 Pattern-matching IDS architecture

Figure 3.2 shows a typical pattern-matching IDS architecture [BACE 00].

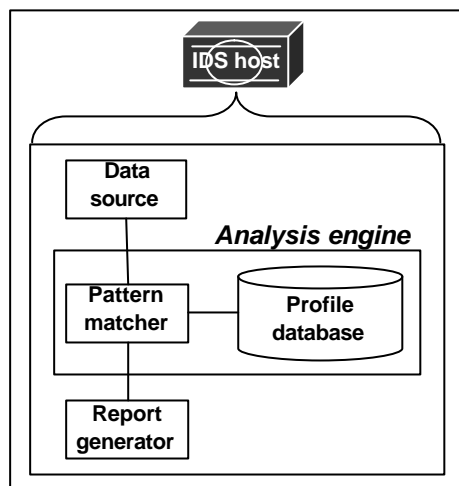


Figure 3.2: Pattern-matching IDS architecture

Pattern-matching IDSs, sometimes referred to as *misuse detection* IDSs [SCHN 00], include the following specific components:

- The **data source** that serves as input to the pattern-matching IDS.
- The **analysis engine** in the architecture, which consists of the following components:
 - The **pattern matcher**, which attempts to detect intrusions by identifying certain patterns of intrusion.
 - A **signature database** that contains known patterns of intrusions.
- The **report generator** for reporting on the intrusions detected.

The **data source** includes anything from operating system audit trails and log files to raw network packets, depending on how the specific pattern-matching IDS is set up to collect source data. Each piece of source data is carefully analysed by the **pattern matcher** and then compared to known intrusion patterns referred to as signatures. These signatures are stored in a **signature database**. The signature database needs to be regularly updated with new intrusion signatures as new intrusion techniques are discovered. When the pattern matcher finds activity that matches a specific signature in its signature database, a **report generator** component compiles a report of the intrusions each time an intrusion is detected. As part of the report generator, alarms may also be sounded for a human to interact on a specific intrusion in progress.

3.2.2.2 Anomaly detection IDS architecture

Figure 3.3 shows a typical anomaly detection IDS architecture [BACE 00].

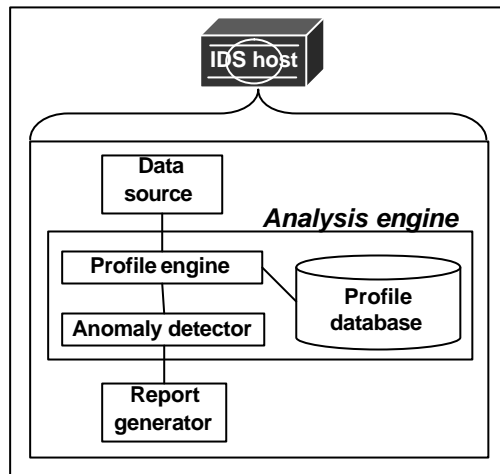


Figure 3.3: Anomaly detection IDS architecture

The **data source** and **report generator** components for the anomaly detection architecture are the same as for the pattern-matching architecture. The *anomaly detection* architecture, however, has the following components that differ from pattern-matching architecture:

- The **analysis engine** in the architecture, which consists of the following components:
 - The **profile engine**.
 - The **anomaly detector**.
- A **signature database** that contains known patterns of *normal* user or system behaviour.

Each piece of source data is carefully grouped by the **profile engine** to form sets of related user or system behaviour. Such a set of behaviour is referred to as a *profile*. A **signature database** contains profiles of *normal* user or system behaviour. The signature database can either be set up manually by a human expert to define profiles, or a computer can be used to compile profiles by using statistical techniques, which can be updated automatically by the computer. The **anomaly detector** then compares each profile compiled from the source data by the profile engine to the normal user

and system behaviour profiles from the signature database. When the anomaly detector finds a profile that appears to be abnormal or unusual compared to a specific user and system profile in the signature database, such behaviour is labelled as intrusive.

Anomaly detection IDSs, however, are difficult to implement, because what is seen as “normal behaviour” for one organisation is not necessarily the same for another. For this reason, most IDSs are based on pattern -matching technology [GRAH 00].

3.2.3 Other approaches to IDS architectures

The sections that follow will take a closer look at IDS architectures that are variations of the distinct IDS architecture approaches . In the literature there are many other approaches to IDS architectures [JAHN 02, KUSP 95, DAVI 01, TRIU 02]. Most of these architectures closely relate to the pattern -matching and anomaly detection architectures . However, the architectures discussed in this section have specifically been selected for discussion because they incorporate the use of interesting techniques .

3.2.3.1 IDML-based intrusion detection

Figure 3.4 displays an architecture for intrusion detection based on intrusion detection markup language (IDML) [LITS 01].

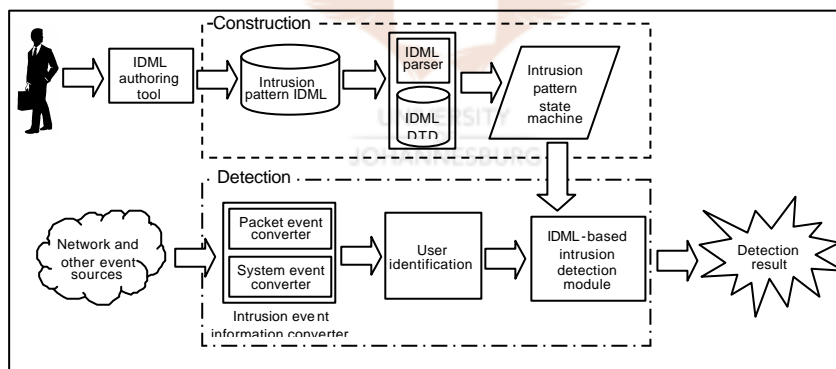


Figure 3.4: An architecture for intrusion detection based on IDML

The main components of the architecture in figure 3.4 are [LITS 01]:

- The construction component.
- The detection component.

The construction component merely uses an XML-based protocol, referred to as IDML, to express intrusion patterns in a computer-processable format. The process for the construction component involves human experts that use an IDML authoring tool to write intrusion pattern IDML documents. The IDML parser is used to validate the intrusion pattern document using the corresponding intrusion pattern, which is stored in a specific format referred to as a document type definition (DTD). If the pattern is valid, the intrusion pattern will be translated into finite intrusion pattern state machines for further use in the detection process.

Almost all intrusion patterns can be transformed into sequences of intrusion actions – an intrusion seldom happens from a single action. Intrusions, therefore, can be represented using a finite intrusion pattern state machine. Various intrusion actions will cause the intrusion process to change from one state to the next, where the state is used to keep track of the current status of the intrusion process. A typical finite intrusion pattern state machine is shown in figure 3.5.

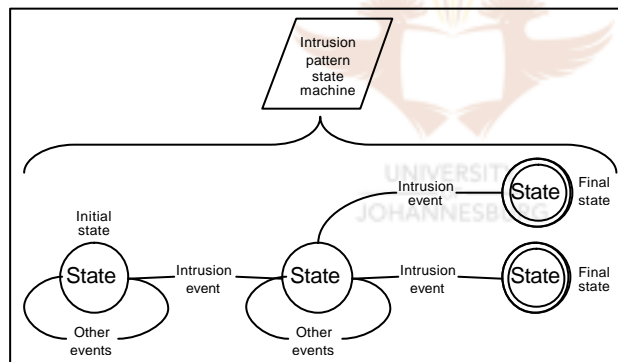


Figure 3.5: A typical finite intrusion pattern state machine

The detection component, on the other hand, incorporates one of the distinctive intrusion detection approaches: pattern matching. It uses network and other event sources, which are converted to packet or system events by an intrusion event

information converter. These events are caused by a specific user account and therefore an attempt is made to retrieve a user's identification on each event in a bid to trace the intrusion to a specific user. This information in conjunction with the IDML-based state information is then used by the IDML-based intrusion detection module to identify and act on intrusions.

IDML-based intrusion detection has some positive and negative sides. On the positive side it attempts to detect intrusions not only by using conventional data sources, for example network traffic and event logs, but also an IDML-based approach which makes the intrusion detection process more successful with fewer false alarms. On the negative side, the number of false alarms is still quite high. In an experiment that was carried out for testing the IDML-based intrusion detection architecture, 25% of all intrusions detected were still false alarms [LITS 01]. Furthermore, IDML-based intrusion detection poses a bigger and more complex processing overhead due to the large number of states that must be tracked by the IDML-based IDS and thus requires additional memory space. This, however, is not too much of a concern since the cost of memory space for a large organisation is not difficult to bridge. Cost, however, is never a factor to be ignored. In addition, attempting to trace intrusions back to a certain user is done by using metadata collected from the data sources, which might only reveal the specific user account being used to launch the intrusion. This may prove to be insignificant information since most of the time an intrusion is launched by using a hacked user account. The IDS, thus, may not be intelligent enough to discover the ID of the real perpetrator.

3.2.3.2 An IDS architecture for detecting TCP SYN flooding

Figure 3.6 displays an intrusion detection architecture for detecting transmission control protocol (TCP) synchronisation (SYN) flooding intrusions [KASA 00].

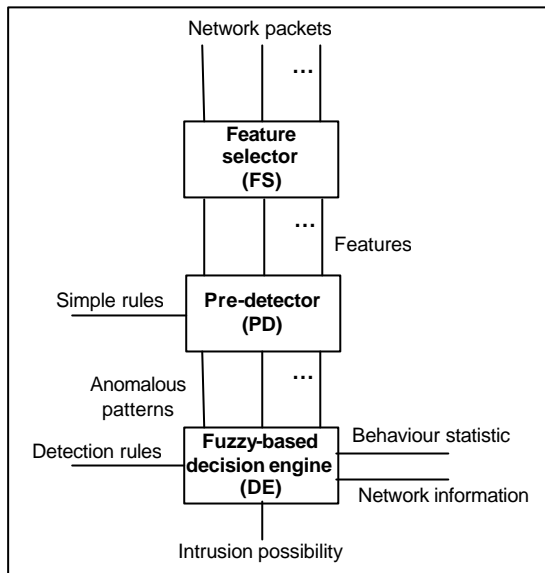


Figure 3.6: An intrusion detection architecture for detecting TCP SYN flooding

The architecture in figure 3.6 is a network-based intrusion detection architecture designed specifically to detect TCP SYN flooding intrusions. This specific intrusion is referred to as a denial-of-service (DoS) intrusion. The intrusion is launched by using TCP to send an excess of SYN data packets over a network to specific systems in an effort to exhaust the network and system resources. The architecture consists mainly of three components:

- The **feature selector (FS)**.
- The **pre-detector (PD)**.
- The **fuzzy-based decision engine (DE)**.

The FS captures packets from the network and extracts certain fields – so-called *features* – from the data packets. The specific features extracted by the FS may not all have exactly the same properties, for example some fields may have different lengths. The PD checks that the selected fields are sorted and all the selected features have the same properties before the DE can detect a possible TCP SYN flooding intrusion.

The positive side of this architecture is that it employs the use of rule-based fuzzy logic when detecting intrusions. Fuzzy logic [YAZA 92] provides a way of creating

more intelligent IDSs. The negative side of this architecture is that it can detect only one specific intrusion. However, there is room for expanding the architecture to be able to detect more intrusions.

These architectures help a lot in finding better IDSs. There are, however, still many problems with IDSs, which are addressed after the following section.

3.2.4 Commercially available IDSs

Examples of IDSs that are commercially available either as freeware or for a price include Snort [SNOR 02], ISS RealSecure [REAL 03], eTrust Intrusion Detection [COMP 03], Network Flight Recorder [NFRS 03], and Cisco IDS [CIDS 03]. Some of these IDSs are able to detect intrusions over multiple operating system platforms, while others can detect intrusions only on specific operating system platforms.

3.2.5 The problems with IDSs

State-of-the-art IDSs, however, fall short in many dimensions [SCHN 00]. They create too many false alarms. If they cry wolf too much, one will stop listening to them. Another problem is that IDSs do not respond to intrusions promptly enough. The main reason for this problem is that they do not have sufficient intelligence to decide in good time what an intrusion is. Furthermore, they fail to intelligently counteract intrusions in an effort to neutralise the intrusion – they normally merely notify and report the intrusion, and then wait for a person to counteract it.

Perhaps the biggest problem with an IDS is the fact that it is a *reactive* information security technology – it does not take preventative measures, but rather attempts to detect an intrusion as soon as it occurs or after it has occurred. Proactive information security technologies thus attempt to smother the problem or prevent an intrusion – before it can occur. One such *proactive* approach is known as *vulnerability scanning* and is discussed below.

3.3 VULNERABILITY SCANNING

3.3.1 What is vulnerability scanning?

The concept of vulnerability scanning is having an automated scanning program, referred to as a vulnerability scanner (VS), that scans a computer or a network of computers for a list of known weaknesses, referred to as vulnerabilities [SCHN 00]. In other words, a VS analyses the security state of a system on the basis of information collected at intervals. After a scan is completed, the VS creates a report of the vulnerabilities found and leaves it up to a person to fix them. Vulnerability scanning is also commonly referred to as *vulnerability analysis* in the industry [BACE 00].

A VS can be seen as a *proactive* information security technology, because it attempts to search for known vulnerabilities *before* the vulnerabilities can be exploited by an intruder. This is done in a very similar way to IDSs, because VSs also use signatures for the vulnerabilities they can identify. Therefore, a VS is an information security technology that is but a special case of intrusion detection [BACE 00]. In addition, an IDS is seen as a *dynamic* information security technology, whereas a VS is seen as a *static* information security technology.

The architecture of VSs is discussed in detail in the sections that follow.

3.3.2 The architecture of VSs

There are some aspects that play an important role in the architecture of VSs. These include the following:

- The **location** of the VS in a network.
- The **scan policy** that specifies the VS setup.
- **Data source** that serves as input to the VS.
- **Analysis engine** that identifies vulnerabilities.
- The **report** that a VS creates.

Some of the above aspects are discussed in detail throughout this section. The scan policy, data source, analysis engine and report aspects of VSs also form part of the main components of a VS. The typical **location** of a VS is shown in figure 3.7 and is essentially the same as for an IDS, except that a VS scans from only one fixed location in the network, and not from multiple locations as an IDS can.

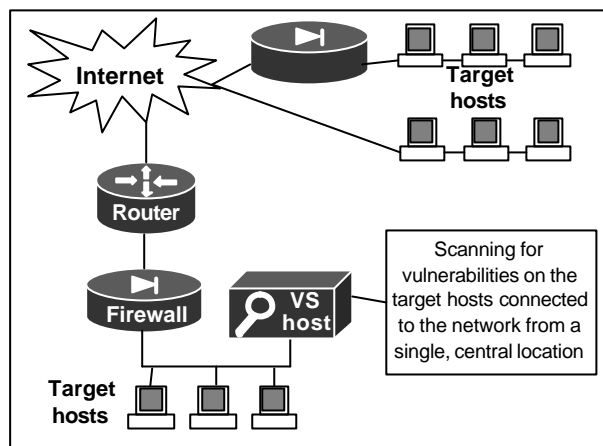


Figure 3.7: The location of a VS in a network

A VS is dependent on a **scan policy** that contains information on how the VS is set up to scan for vulnerabilities. This scan policy is usually reconfigured for each specific scan. In contrast, an IDS continually monitors data sources for all possible intrusions that it is able to detect. For example, a VS's scan policy may be set up to scan only selected hosts on a network. In addition, it may also be set up to scan for only specific types of vulnerabilities logically grouped into specific categories of vulnerabilities. The reason for scanning only for certain categories of vulnerabilities is to save network and system resources when these resources are critically depended on for purposes other than vulnerability scanning, because VSs can sometimes exhaust these resources when scanning and testing for denial-of-service vulnerabilities [MCSK 02], for example. On the other hand, IDSs need to monitor for all possible intrusions in real time, and therefore should not detect only a subset of intrusions. If the detection of some intrusions is omitted, the IDS might miss the detection of a possible intrusion and this will defeat the purpose of an IDS.

VSs also collect **source data** which will be interpreted by a dedicated *VS host* in a bid to find vulnerabilities. There are two different levels at which a VS can scan for vulnerabilities, namely host level or application level. The different levels of data sources that a VS can scan are referred to as the *targets*. IDSs detect intrusions on four different types of targets, as discussed earlier in this chapter, namely network-based targets, host-based targets, application-based targets, and target-based targets.

VSSs, however, only scan for vulnerabilities on two of those four targets. The following are the types of targets that a VS can scan for vulnerabilities [COLE 02]:

- *Host-based targets*: The target here is an individual host. The VS scans the host's configuration settings, typically at operating system level, looking for vulnerabilities.
- *Application-based targets*: The target here is one or more specific applications that are running on a target host. The VS scans the configuration settings for the specific application, looking for vulnerabilities.

The **analysis engine** compares the source data with a predefined known set of data configurations. The analysis engine is also commonly referred to as a *vulnerability matcher* in VS terminology. If the source data contains a specific data string that is also found in the known set of data configurations, then a vulnerability is found or matched. A detailed **report** is produced after the entire scan process is complete.

The architecture on which VSSs are based is derived from IDSs. Vulnerability scanning is a special case of intrusion detection. This means that VSSs partly employ one of the two distinctive architectures of IDSs, namely pattern matching. The only difference here, however, is that IDSs attempt to *match* a set of actions, which occurred in a specific sequence, to a pattern to find an intrusion. VSSs, on the other hand, attempt to *match* only a specific string of data to a known signature of data to find a vulnerability. An architecture for VSSs is shown in figure 3.8 [BACE 00].

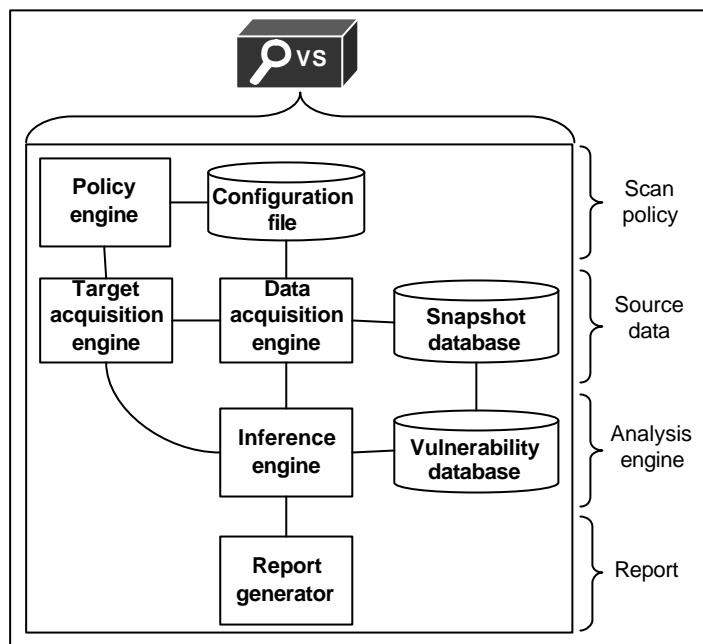


Figure 3.8: An architecture for a VS

The architecture of VSs consists of the following components:

- The **scan policy** component contains the following two sub-components:
 - **Policy engine**: Loads or stores the scan configuration as the user set it up.
 - **Configuration file**: Contains the scan configuration, i.e. settings and options of the VSs as specified by a user. An example of such a setting is the IP address range of systems to be scanned.
- The **source data** component contains the following three sub-components:
 - **Target acquisition engine**: Searches for the specific target hosts to determine whether a host is online or not.
 - **Data acquisition engine**: Samples the systems' attributes and configuration and stores them in the snapshot database.
 - **Snapshot database**: Contains the target hosts' characteristics and configuration as collected by the data acquisition engine.
- The **analysis engine** component contains the following two sub-components:
 - **Inference engine**: Controls the target and data acquisition engines, and matches the snapshot database with the vulnerability database to detect which vulnerabilities are apparent in the systems that were scanned.

- **Vulnerability database:** Contains the signatures of all known weaknesses in software or hardware.
- The **report generator** of the VS creates a report that contains a detailed description of the signatures that matched between the snapshot database and the vulnerability database, which are the vulnerabilities detected by the VS. A VS report usually also contains more information on how and where to fix the vulnerabilities that were found.

Apart from the architecture of VSs as discussed above, there are other approaches in the literature. One such other approach is discussed in the section that follows.

3.3.3 Another approach to VS architectures

There are currently not many approaches to VS architectures other than the one discussed in the previous section. However, the following is a VS architecture that follows a more decentralised approach.

Figure 3.9 displays an architecture for distributed vulnerability scanning [LOPY 01].

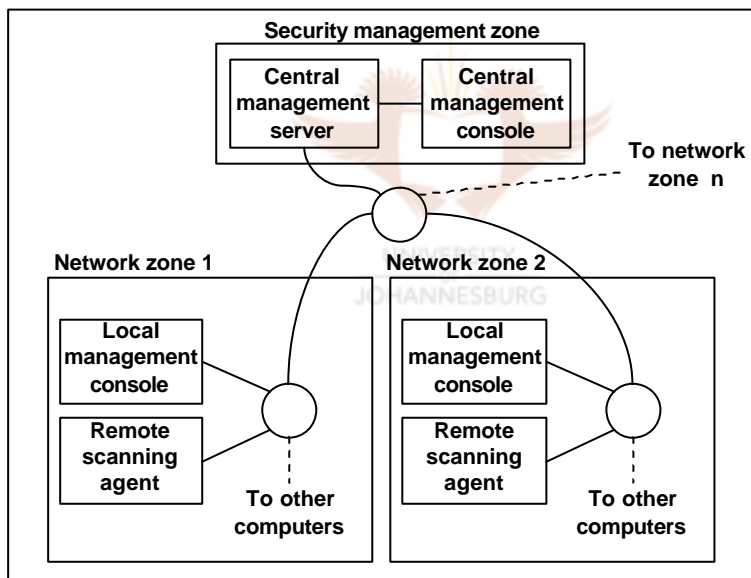


Figure 3.9: A distributed architecture for vulnerability scanning

The architecture in figure 3.9 shows a network that contains a security management zone and subnet zones with the following main components:

- Security management zone containing:
 - The **central management server**.
 - The **central management console**.
- Different subnet zones, each containing:
 - A **local management console**.
 - A **remote scanning agent**.

A **central management server** and a **central management console** form the security management zone. The central management console is used as a front-end manager to the central management server. The central management server conducts all control operations, schedules scanning tasks, maintains the security policy, updates scanning modules, and delivers them to a remote agent on demand or by schedule. Each subnet zone has a **remote scanning agent** and a **local management console**. The remote scanning agent receives commands, procedures, and schedules from the central management server to scan the specific network. These commands, procedures, and schedules can also be received from the local management console for performing certain decentralised actions.

The workload that VSs create when conducting a scan is normally very high and a multitude of system resources are occupied. The positive side of this architecture, however, is that it has multiple scanning agents, each situated in its own subnet. The workload in the case of having a single server that has to scan all subnets can drain the entire network resources significantly. Having multiple scanning agents thus reduces the utilisation of system resources significantly. The negative side of this architecture is that it is much more expensive. In addition, this architecture of vulnerability scanning offers no intelligent scanning techniques.

3.3.4 Commercially available VSs

Examples of VSs that are commercially available either as freeware or commercial software include CyberCop Scanner [CYBE 03], Cisco Secure Scanner [CSSC 03], Nessus [DERA 03], Internet Scanner® [ISSC 03], SAINT [SAIN 03], and NetRecon [NETR 02].

3.3.5 The problems with VSs

VSs work in a strange and unorthodox way: they perform a scan by attempting to break through the current security features on a computer. The question could be asked why one would use a VS if it damages the security on the computer. However, this is not exactly true: the VS does not really damage the security on a computer, but simulates and generates “watered-down” or “fake” attacks on the security of a computer to find out if a computer might be flawed in such an attack if the attack were launched by a hacker. It is exactly these “simulated” attacks that can drain the network resources, forcing the network to its knees, or completely killing a network.

In the light of the above fake attacks, VSs sometimes have to make assumptions on the way a specific computer reacted to a fake attack, since launching a full-fledged attack might cause real damage to a computer and/or the network. Making such assumptions can be very dangerous since it may be difficult to tell whether a full-fledged attack was successful or not. It is for this reason that some VSs today indeed launch full-fledged attacks, but – as just mentioned – it might cause damage to a computer. Therefore, backups should be made before the scan is conducted and it should be remembered that conducting a scan takes up valuable time and system resources.

VSs all utilise some sort of database with the same goal: to store the signatures of the vulnerabilities they can detect when they scan for the vulnerabilities. A major problem with these VS databases, however, is that they are disparate in the specific way that the vulnerabilities are named and organised in the vulnerability database of each different VS. This disparity is caused mainly by the difference in structure of almost any VS's vulnerability database.

For example, some VSs store hundreds of vulnerabilities in their vulnerability databases simply sorted from *vulnerability 1* to *vulnerability n*. The problem with this database structure is that the vulnerabilities are not organised, for example, related vulnerabilities are not grouped together. In addition, different VSs that employ this database structure may name the same vulnerability in different ways. For example, one VS might call a particular vulnerability as “a Trojan horse”, while

another might refer to the same vulnerability as “a backdoor”, and yet another might refer to it as “a virus” where these names have the same meaning.

VSs group certain vulnerabilities together to form different vulnerability categories. A vulnerability category refers to the grouping of specifically the same types of vulnerabilities, in other words vulnerabilities with the same genre of characteristics. Another database structure disparity example is that different VSs address different vulnerability categories. In other words, vulnerabilities that are grouped in a particular vulnerability category by a specific VS might be grouped in a different vulnerability category by another VS. One VS might group a vulnerability, for example “a remote share was found without any password defined”, in the *password guessing and grinding* vulnerability category, while another VS might group this vulnerability in the *remote access & services* vulnerability category.

What is more, some VSs define a small number of vulnerability categories, while other VSs define many vulnerability categories. Different VSs might even address the same kind of vulnerability in a different way, for example one VS might audit passwords by using a dictionary-attack technique, whereas another might do so by using a brute-force-attack technique. Disparity in the database structure is a major problem, especially when choosing a specific VS to use in an organisation.

3.4 CONCLUSION

IDSs and VSs are both information security technologies that enhance the security on a computer and network in that they detect and prevent intrusions and attacks from happening, respectively, with a relatively good success rate. IDSs and VSs, however, still produce many problems and challenges for future research. There is a good possibility that hybrid systems might be seen in the future – that is, programs that incorporate IDS and VS technologies in one system [MOHA 01]. One should refrain, however, from running an IDS tool and a VS tool in parallel in the same environment because when a VS attempts to cast a “simulated attack” on designated hosts, an IDS running in the same environment will identify such a simulated attack as a real intrusion and will increase the false alarm rate of the IDS in due course.

Although the cost difference between IDSs and VSs is not a predominant factor, it is interesting to note that the overall cost of implementing and maintaining VSs is higher than that of IDSs [ESCI 02].

It is generally better, though, to follow a proactive approach than a reactive approach because prevention is better than cure. It is for these reasons that VSs will be used rather than IDSs as part of the model for vulnerability forecasting introduced later in this thesis. The problem, however, is that VSs are different software products, which scan for different types or categories of vulnerabilities. There is a need, thus, to create a “standardised” set of vulnerability categories which will enable the vulnerability forecasting model to use any VS, or even a multiple of VSs. This method of standardising vulnerability categories is referred to as **harmonised vulnerability categories**, which is discussed in the next chapter.



CHAPTER 4

HARMONISING VULNERABILITY CATEGORIES

4.1 INTRODUCTION

A major problem with VS databases, as discussed in the previous chapter, is that they are disparate in the specific way that the vulnerabilities are named and organised in the vulnerability database of each different VS. This problem might be resolved by having harmonised vulnerability categories. These categories should cover the full scope of potential vulnerabilities. The aim of having harmonised vulnerability categories is to have a measure onto which the vulnerability categories of any VS can be mapped to determine the level of vulnerability category competence for each specific VS. This specific problem is addressed in this chapter.

In the remainder of this chapter, the concept of harmonising different sets of vulnerabilities into **harmonised vulnerability categories** is introduced, followed by a discussion of each category with examples to demonstrate the usefulness of the proposed categories.

4.2 METHOD OF IDENTIFYING CATEGORIES

A major problem with VS tools is that they sometimes attempt to address an excessively wide variety of vulnerabilities. As mentioned in the previous chapter, the specific vulnerabilities that VS tools check for, however, differ significantly from tool to tool. Using only one specific VS tool may prove to be insufficient in scanning for certain types of vulnerabilities. For example, CyberCop Scanner [CYBE 02] scans extensively for vulnerabilities of the type *misconfigurations*, whereas Cisco Secure Scanner [CSSC 00] gives minimum attention to misconfiguration vulnerabilities. Furthermore, different VS tools sometimes refer differently to the same vulnerability. For example, CyberCop Scanner refers to *mail transfer* and Cisco Secure Scanner to

SMTP, which is, in essence, the same set of vulnerabilities. How will the results of a vulnerability scan done by a specific tool, e.g. CyberCop Scanner, compare with those of another, e.g. Cisco Secure Scanner? To answer this question, a **common** set of vulnerabilities is required. The researcher proposes such a common set of vulnerabilities, which was determined by evaluating a number of different sets of vulnerabilities. This common set of vulnerabilities will be referred to as a “harmonised” set of vulnerability categories.

The harmonised vulnerability categories were identified by analysing the Internet security vulnerabilities as found in literature [NOCF 01] [BACE 00] [SCMK 01] [GREE 02] [NORT 01] [KEOS 01], as well as those used by popular VS tools such as CyberCop Scanner and Cisco Secure Scanner. The criteria for identifying the harmonised vulnerability categories were based on the following [BISH 99]:














- Vulnerabilities of a **similar nature** should be grouped together.
- Classification should **not be based on the social cause** of the vulnerability. This includes issues such as *motive, intent, and malicious or accidental cause*.

The researcher identified 13 harmonised vulnerability categories. These categories are discussed in the section that follows.

4.3 HARMONISED VULNERABILITY CATEGORIES

A harmonised vulnerability category represents a certain group or class of vulnerabilities, which have the same genre of vulnerability characteristics. For example, all vulnerabilities related to compromising passwords, such as “a password is a dictionary word” or “a password is shorter than 8 characters” or “a password is sent in clear text”, can form a harmonised vulnerability category called *password cracking and sniffing*. It is well known that VS tools in the industry represent solutions for rectifying vulnerabilities as well. It should be mentioned that the rectification of vulnerabilities is beyond the scope of this chapter. In other words, the purpose of this chapter is to identify harmonised vulnerability categories only, and not to present solutions for various vulnerabilities. Before discussing each harmonised vulnerability category in detail, a summary of the categories is given in table 4.1.

Table 4.1: Summary of the harmonised vulnerability categories

Harmonised vulnerability category	Brief description
1  Password cracking and sniffing	Vulnerabilities with a root cause of having accounts with weak or no passwords
2  Network and system information gathering	Vulnerabilities concerned with scanning a network to discover a map of available hosts and vulnerable services
3  User enumeration and information gathering	Vulnerabilities concerned with retrieving information of user accounts from a specific system
4  Backdoors, Trojans and remote controlling	Vulnerabilities concerned with having hidden access mechanisms installed on a system
5  Unauthorised access to remote connections & services	Vulnerabilities concerned with the risk that an unauthorised person has the ability to connect to and misuse a system
6  Privilege and user escalation	Vulnerabilities concerned with the risk that the access rights of an existing user account can be upgraded by an unauthorised user, granting more privileges to the user
7  Spoofing or masquerading	Vulnerabilities concerned with the risk that an intruder can fake an IP address in a bid to act as another person
8  Misconfigurations	Vulnerabilities concerned with the risk that applications have been incorrectly configured
9  Denial-of-services (DoS) and buffer overflows	Vulnerabilities concerned with the risk of one or more intruders launching an attack designed to disrupt or deny legitimate users' or applications' ability to access resources
10  Viruses and worms	Vulnerabilities concerned with malicious programs
11  Hardware specific	Vulnerabilities concerned with having hardware peripherals that execute ROM-based or firmware-based programs
12  Software specific and updates	Vulnerabilities concerned with the risk that specific software applications contain specific, well-known bugs
13  Security policy violations	Vulnerabilities concerned with the risk that an Internet security policy has been violated

4.3.1 Password cracking and sniffing

This category involves vulnerabilities with a root cause of having accounts with weak or no passwords. Tools are readily available on the Internet that can be used to intercept passwords from any transmission over the Internet. These kinds of tools are better known as *sniffers*.

On some systems, passwords are stored in cleartext, or transmitted in cleartext over the Internet. If an attacker manages to intercept cleartext passwords, the passwords do not even need to be cracked. To solve this problem, passwords are transmitted or stored on a system in encrypted format. Still, it is possible to sniff these encrypted passwords from the Internet and then use password-cracking tools, for example L0pht Crack [LOPH 02], to crack the passwords. Given that a user has administrative access, L0pht Crack can also retrieve the *stored* encrypted passwords on a system in an attempt to crack them.

Examples of vulnerabilities belonging to this category are the following:

- If the FTP service is enabled, anyone can try to guess passwords to connect to the FTP service.
- A malicious user could remotely retrieve the system's password file. This can lead to further system access, including administrator access.

4.3.2 Network and system information gathering

This category involves vulnerabilities concerned with scanning a network to discover a map of the available hosts, as well as to detect vulnerable services on the hosts and the network. Furthermore, these vulnerabilities get information on the hosts found on the network to determine the specific hardware or software applications used.

Having a map of a network and information on which software applications are used in an organisation may help an intruder to gain sufficient information on the target and to determine which specific hacking techniques to use. *Footprinting, network mapping, target acquisition, and network reconnaissance* are synonyms found in the literature [SCMK 01] [NORT 01] for network and system information gathering.

Examples of vulnerabilities belonging to this category are the following:

- The routing table could be retrieved, which reveals information of the physical network setup.
- Using the FTP SYST command, attackers can discover operating system version information. This can lead to administrator access and malicious activity.

4.3.3 User enumeration and information gathering

This category involves vulnerabilities concerned with retrieving information of user accounts from a specific system, for example the user account name (e.g. *bretl*) and the user details (e.g. *Bret Lee, General Manager, Office 227, Accounts Department*).

An attacker can use this information typically to identify that Bret Lee is a general manager, whose computer could contain more sensitive information than a normal employee's computer, making the manager's computer a more sought-after target. Furthermore, as soon as an intruder has retrieved a list of the user account names

registered on a specific system, it is often only a matter of time before he/she obtains the password by using a password-cracking program, for example L0pht Crack [LOPH 02]. After all, the user account names have to be obtained before any attempt can be made to crack passwords.

Examples of vulnerabilities belonging to this category are the following:

- Using the “finger” command on a specific system will retrieve a list of all the user account names on that system.
- Null session connections can be used by an attacker to list sensitive user account information, such as revealing the identity of a user on the system.

4.3.4 Backdoors, Trojans and remote controlling

This category involves vulnerabilities concerned with having access mechanisms installed on a system which are almost hidden and not obvious. In other words, a covert channel is created.

Often a backdoor is installed with the aim of controlling a system remotely. The backdoor becomes a hidden entry point where the intruder can connect to the system unnoticed at any given time. Most of the time, the “vehicle” for establishing such backdoors is called a “Trojan horse” or a “Trojan” [SCMK 01]. A Trojan is a software application that operates under the impression that it is intended for a specific purpose, but actually performs hidden operations as well. For example, most of the time Trojans are sent to someone as an e-mail attachment in the form of, for example, a game. As soon as the person opens that attachment, the game can be played successfully while a backdoor is unknowingly created in the background by the game.

Examples of vulnerabilities belonging to this category are the following:

- Back Orifice [BACK 02] or Netbus (recently called Spector) [NETB 02] are Trojan horse programs that, as soon as they are installed on a system, create backdoors, enabling remote controlling of the system.
- Remote controlling software is installed on the system, but it is not password-protected, allowing anyone to remotely connect and take over the system.

4.3.5 Unauthorised access to remote connections and services

This category involves vulnerabilities concerned with the risk that an unauthorised person has the ability to remotely connect to a system via a specific port with the aim of misusing the system.

Gaining access to remote connections and services is often used in an attempt to exploit more vulnerabilities, since gaining this access will “open more doors” to other vulnerabilities. For example, if the TELNET service is running, anyone can attempt to connect to, for example, a guest account. Connecting to the TELNET service itself can do no harm. An attacker, however, can now gain information on the particular operating system that runs the TELNET service. This could lead to additional malicious activity by the attacker.

Examples of vulnerabilities belonging to this category are the following:

- An attacker could use an anonymous FTP server to launch exploits against another system to gain special access. An attacker could use this special access to possibly bypass firewalls.
- After anonymous access to the FTP server has been gained, the attacker can try to exploit further vulnerabilities in the FTP service, for example to see if the FTP root directory is write-enabled in a bid to store unauthorised data or information.

4.3.6 Privilege and user escalation

This category involves vulnerabilities concerned with the risk that the authorisation properties of an existing (probably compromised) system account can be changed so that this user account has more privileges or more powerful access rights allocated to it than was initially intended.

More privileges and more powerful access rights will allow a specific user account to access data or system resources in an effort to access specific data or information that was previously inaccessible to the user account. For example, an account with

standard user rights might have been escalated to an account with administrative rights.

Examples of vulnerabilities belonging to this category are the following:

- An attacker could execute arbitrary commands remotely as the user who is running the HTTP server. If the owner of the HTTP server has administrative access, the attacker can remotely execute commands as an administrator.
- Some registry entries on a Windows system may be remotely accessible, allowing the modification of the permissions of these registry entries.

4.3.7 Spoofing or masquerading

This category involves vulnerabilities concerned with the risk that an IP packet's source address can be faked to hide an intruder's identity or activity amongst a storm of other network traffic.

For example, assume *network A* is protected by a firewall that only allows IP addresses with source addresses in the subnet mask of 123.213.44.0. Assume an attacker is sitting in *network B* with a subnet mask of 211.143.2.0. The attacker could now create a packet in *network B*, which will have a source address of, for example, 211.143.2.67. By using the appropriate spoofing tool, the attacker can now easily change this source address to, for example, 123.312.44.67. The firewall in *network A* will now allow the packet created by the attacker through into *network A*.

Examples of vulnerabilities belonging to this category are the following:

- If a poorly configured firewall is installed, attackers can launch attacks using the identity of the firewall server, thus masking their true identity. If any hosts or networks allow special access to this server, then the attacker has the same access.
- IP forwarding is found to be enabled, allowing the host to act as a router so that other hosts can forward packets through this host. If this host is running a firewall, then the firewall can be bypassed using IP forwarding.

4.3.8 Misconfigurations

This category involves vulnerabilities concerned with the risk that applications have been incorrectly configured, leaving these applications vulnerable to several of the other harmonised vulnerability categories mentioned here.

Misconfiguration vulnerabilities mostly tend to occur after the installation of new software, because new software is always installed with *default* configuration settings. It is of the utmost importance that newly installed software be reconfigured immediately after installation. In addition, the new configurations must be tested to make sure that they are *correct* and not *misconfigured*.

Examples of vulnerabilities belonging to this category are the following:

- If anonymous FTP is not configured securely, an attacker may be able to perform reconnaissance, delete or modify files, or use anonymous FTP as a distribution mechanism for unwanted files, such as pornography or pirated software.
- If permissions are incorrectly set in the Windows registry to “Everyone”, an attacker could gain access to the registry and commence with arbitrary attacks.

4.3.9 Denial-of-services (DoS) and buffer overflows

This category involves vulnerabilities concerned with the risk of one or more intruders launching an attack designed to disrupt or completely deny legitimate users' access to networks, servers, services, or other resources.

DoS vulnerabilities are not concerned with stealing information or changing data, but simply with downgrading the performance of the computer and/or network resources to such a level that services are disrupted significantly or completely. Consider an online shop that is completely reliant on the Internet to conduct business. Suppose an attacker manages to fill up the storage space of the online shop's servers by uploading junk data to it. This can potentially cause the servers to crash. It could take hours or perhaps days to sort out and restore the servers again, causing the online shop to lose so much money that it might have to close down.

Examples of vulnerabilities belonging to this category are the following:

- An attacker can create files on the hard disk of the Web server and fill it up, leaving the service of the hard disk interrupted and unavailable.
- An out-of-band data attack can consume all memory and cause a system to reboot. This attack could also cause a system to be unable to handle network traffic. The only way to recover is to either reset or reboot the system.

4.3.10 Viruses and worms

Viruses and worms are different types of software applications, but with the same goal of spreading from one system to another to conduct malicious activity.

Viruses and worms can be considered as some of the most active and malicious vulnerabilities that can be found on a system. Unfortunately, this is the vulnerability category that is often completely neglected by IDSs. Almost any new virus that appears on the Internet scene these days causes havoc all over the world in a matter of hours. The reason is that they all spread through the Internet, be it through e-mail messages, or through vulnerabilities exploited in networking services. For example, if an IDS could also detect for viruses and worms, the famous Code Red and Code Blue worms [HANC 01] would never have caused such havoc around the world in such a short time – they infected systems around the world in less than a day by spreading through an exploit in well-known Web servers all over the world. It should be mentioned that it becomes evident that this problem is addressed in the latest *reactive* IDSs.

Examples of vulnerabilities belonging to this category are the following:

- An e-mail attachment is opened without it first being scanned by a virus detection program. This might allow a virus to infect the system.
- Certain updates or patches are not installed for the Web server, making the server susceptible to a denial-of-service attack.

4.3.11 Hardware specific

This category involves vulnerabilities concerned with having hardware peripherals which do not run software applications, but which rather run ROM-based or

firmware-based programs. These peripherals also contain exploits that cannot be easily updated, patched or corrected, except if the hardware is physically replaced or the firmware is updated.

Examples of such hardware peripherals are network switches, routers and terminals. The main reason why updating the firmware of these hardware peripherals is often neglected is that the peripherals do not have dedicated *owners* as opposed to a computer workstation which has one or more specific dedicated owners. Often the system administrator alone has to see to all of these peripherals in a network. Chances are better for an attacker to discover and exploit vulnerabilities on these peripherals before the administrator will discover that irregularities are happening on them.

Examples of vulnerabilities belonging to this category are the following:

- An attacker can cause a router or switch device to crash and reload. Possible loss of configuration information may result as a consequence of this attack.
- A shared printer may be found on the network without having any authentication enabled on it, leaving it open to a variety of possible attacks. For example, some modern printers host a complete operating system on them. A network printer is often considered as highly trusted and trust relationships are set up accordingly as “wide open”. If access to the operating system of such a printer is gained, an attacker can gain access to all those systems connected to the printer.

4.3.12 Software specific and updates

This category involves vulnerabilities concerned with the risk that specific software applications contain specific, well-known bugs. Because these bugs or exploits are published widely on the Internet [BUGT 02], anyone, including an attacker, is able to access the Internet and collect information about these bugs to try and exploit them.

Software applications must be updated to *patch* their exploitations in an effort to fix security bugs or loopholes to avoid successful future attacks on them. For example, recently there have been enormous denial-of-service attacks on Microsoft’s Internet

Information Server by the very famous Code Red and Code Blue worms [SECF 02]. Therefore, Microsoft had to make *software patches* available to fix the vulnerabilities that were exploited so lustily by these Internet worms.

Examples of vulnerabilities belonging to this category are the following:

- A service pack installed is outdated. Vulnerabilities discovered after the specific service pack was installed on this system leave a potential threat unless they are patched by the latest service pack.
- An insecure logon method is allowed for a Web server, causing a threat that a user name and password may be sniffed through this method.

4.3.13 Security policy violations

This category involves vulnerabilities concerned with the risk that an Internet security policy has been violated. An Internet security policy is a set of security rules created internally by an organisation. It can specify how systems in the organisation should be configured to be on a security level that is acceptable for the organisation. One of the policy statements might specify, for example, that the user's password will expire every 30 days.

When a security policy violation is found, it means that a different configuration setting on the system was detected and thus violates the prescribed policy setting. It is of the utmost importance, though, that management specify the security policy *correctly before* it is implemented electronically. The policy might be implemented correctly according to the policy document, but if the document specification is wrong, its electronic implementation will also be wrong!

Examples of vulnerabilities belonging to this category are the following:

- The system's event or security log is not restricted according to the system's security policy. Anyone will thus be able to alter or delete the logs.
- The system's screensaver lockout is not enabled according to the system's security policy and will not automatically lock the system if the owner of the system neglected to lock the system himself/herself.

4.4 STANDARDISATION OF VULNERABILITIES

After this research was initiated, a similar initiative evolved on the Web in which a common standard for the naming of vulnerabilities was introduced. This standard is referred to as the common vulnerabilities and exposures (CVE) standard [MITR 03].

CVE is a list or dictionary that provides common names for publicly known information security vulnerabilities and exposures. Using a common name makes it easier to share data across separate VS databases. While CVE may make it easier to search for common vulnerabilities, it should not be considered as a vulnerability database on its own merit, because it is only a common reference to the same vulnerabilities addressed by different VSs and may not necessarily be an exhaustive list of all possible vulnerabilities.

In addition, CVE does not provide for harmonised vulnerability categories as discussed in this chapter. CVE provides a method of referencing the same vulnerabilities in different VSs only. Harmonised vulnerability categories, however, attempt to provide a method of referencing the same *categories* of vulnerabilities for different VSs. In other words, where CVE attempts to standardise the naming of vulnerabilities across different VSs, harmonised vulnerability categories attempt to standardise the categorisation of the same vulnerability *categories* across different VSs.

4.5 CONCLUSION

The harmonised vulnerability categories can serve as a useful management tool. These categories reflect all vulnerabilities in state-of-the-art VSs today as well as those vulnerabilities found in current literature. The 13 harmonised categories will serve as generic categories for categorising vulnerabilities found in state-of-the-art VS tools. They will expand and evolve along with the evolution of information technology and its applications.

Be that as it may, such a construction of harmonised vulnerability categories will contribute significantly to safer and better managed Internet information security in

terms of providing a mechanism that can be used as a measure for identifying how different VS products comply with “standardised” vulnerability categories referred to as harmonised vulnerability categories. The next chapter will demonstrate how harmonised vulnerability categories can be used in order to find a way in which to refer to the same vulnerability *categories* across different VS products.





CHAPTER 5

VULNERABILITY SCANNER PRODUCTS

5.1 INTRODUCTION

Due to the increasing awareness of the public of security issues on the Internet, there are a myriad of security products available on the software market today and this number is increasing. Hence the dilemma when choosing the right security product for a particular organisation's security needs.

The focus of this chapter is to develop a better understanding of state-of-the-art VS products. There are many VS products available on the software market. As was pointed out in previous chapters, they often refer to the same vulnerability in a different way and this makes it very difficult to see exactly which vulnerabilities are scanned for by the different VS products. This dilemma can be solved by using the framework of **harmonised vulnerability categories** [VEE2 03] as shown in the previous chapter in table 4.1. Other aspects of VS products are also considered in this chapter, for example the specific database structure of a VS, in an attempt to shed more light on the problems that the different VS products pose.

The sections that follow will discuss VS products in more detail. An overview of the state-of-the-art VS products is given. Some of these products are discussed in detail, with the emphasis on the databases that they employ.

5.2 VS PRODUCTS

It is important to be aware of the different VS products available on the software market before studying some of them in more detail. There are freeware as well as commercial versions of VS products available and some products differ extensively from others. The section that follows lists some of the major role players in VS

technology and attempts to place the different aspects of the products in perspective to each other.

5.2.1 VS product overview

The VS products discussed in this chapter are the best-known VS products available on the software market today. Table 5.1 shows a list of some of these VS products.

Table 5.1: State-of-the-art VS products

VS product	Commercial or freeware	Reference
bv-Control	Commercial	[BIND 03]
Cisco Secure Scanner	Commercial	[CSSC 03]
CyberCop Scanner 5.5	Commercial	[NETW 03]
Internet Security Scanner (ISS) 6.2.1	Commercial	[ISSN 03]
Nessus Security Scanner	Freeware	[DERA 03]
NetRecon 3.5	Commercial	[SYMA 03]
Nmap 2.5	Freeware	[INSE 03]
Retina 4.7	Commercial	[EEYE 03]
Security Administrator's Integrated Network Tool (SAINT) 4.0	Commercial	[SAIN 03]
Security Analyzer 5.1	Commercial	[NETI 03]
STAT Scanner Professional	Commercial	[HARR 03]

The CyberCop Scanner, the Cisco Secure Scanner, the SAINT, the ISS, and the Nessus Security Scanner will be discussed in more detail in the following five sections. The focus of the discussion of these products will not be to evaluate and compare them with each other, but rather to comment on the practical experience encountered by the researcher while working with the products. This is followed by elaborative discussions on each product's vulnerability database in terms of differences.

5.2.2 CyberCop Scanner

The CyberCop Scanner version 5.5 is discussed because it is well known and widely used for vulnerability scanning today. The creators of the CyberCop Scanner recently decided to replace their CyberCop Scanner VS product with a Web-based product known as the CyberCop ASAP [MCAF 03]. A trial version of the current CyberCop Scanner software is still available for evaluation purposes.

5.2.2.1 Practical experience with the CyberCop Scanner

The CyberCop Scanner was installed on a Windows workstation and then set up to scan workstations, servers, hubs and switches connected to the network for the vulnerabilities as specified in its vulnerability database. Depending on the size of the

network, the CyberCop Scanner scans the network for several hours before the scan is complete. It then generates a report of several hundred pages. Figure 5.1 shows an extract of one of the vulnerabilities in this report.

Vulnerability ID	30006
Description	Remote Access Service (RAS) detected on the host. RAS lets remote users use a telephone line and a modem to dial into a RAS server and use the resources of its network.
Security concerns	A person could be using RAS to gain access to a network from a remote location. This essentially creates a "backdoor" into a network which can bypass the network's firewall, for example.
Rectification procedures	Investigate this host to identify if it is indeed an approved RAS host. If it is an approved RAS host, there may be ways to further secure the host. E.g., RAS can be configured to establish a connection only by automatically calling a user back. This ensures the telephone number of the user that is gaining access via this RAS host is known by the RAS server.

Figure 5.1: An extract from the CyberCop Scanner report

An advantage of the CyberCop Scanner report is that it contains good and detailed description and rectification procedures. However, this report has some disadvantages. It is too long and will take days to study. It is also very technical and requires skilled human resources to rectify the vulnerabilities. The report also does not prioritise the vulnerabilities detected. Another disadvantage is that the CyberCop Scanner is not CVE-referenced.














5.2.2.2 CyberCop Scanner vulnerability database

Of the 13 harmonised vulnerability categories, categories 3, 4, 7, 10 and 11 are covered in very little detail, if at all, by the CyberCop Scanner's vulnerability database, as shown in table 5.2.

5.2.3 Cisco Secure Scanner

The Cisco Secure Scanner version 2.0 [CSSC 03] is discussed because this scanner is probably the most renowned and established networking hardware manufacturer today. The creators of the Cisco Secure Scanner, however, recently announced that this product had reached end-of-life status [CEOS 03] and would no longer be available for sale. Nevertheless, the Cisco Secure Scanner was still chosen for discussion since it can run on multiple operating systems, scan for vulnerabilities on multiple operating systems and will still be supported by the Cisco Secure Scanner for a limited period.

Table 5.2: Harmonised vulnerability categories covered by CyberCop Scanner

Harmonised vulnerability category	CyberCop Scanner
1  Password cracking and sniffing	✓
2  Network and system information gathering	✓
3  User enumeration and information gathering	✗
4  Backdoors, Trojans and remote controlling	✗
5  Unauthorised access to remote connections & services	✓
6  Privilege and user escalation	✓
7  Spoofing or masquerading	✗
8  Misconfigurations	✓
9  Denial-of-services (DoS) and buffer overflows	✓
10  Viruses and worms	✗
11  Hardware specific	✗
12  Software specific and updates	✓
13  Security policy violations	✓

5.2.3.1 Practical experience with the Cisco Secure Scanner

The Cisco Secure Scanner was installed on a Windows workstation and then set up to scan workstations and servers connected to the network for the vulnerabilities as specified in its vulnerability database. The Cisco Secure Scanner can run on Windows as well as on UNIX operating systems. Depending on the size of the network, the Cisco Secure Scanner scans the network for several hours before the scan completes and a large report is generated. Figure 5.2 shows an extract of one of the vulnerabilities in this report.

One advantage of the Cisco Secure Scanner report is that it contains good and detailed description, consequences, and countermeasure procedures. The disadvantage of this report is that it requires effort to work through because of its size. Another disadvantage is that the Cisco Secure Scanner is not CVE-referenced.

FTP Directory and File Permissions

Description

File Transfer Protocol (FTP) is one protocol by which files can be transferred to and from remote computer systems. The user transferring a file usually needs authority to login and access files on the remote system.

Consequences

A remote attacker may be able to perform reconnaissance, delete or modify files, or use the FTP server as a distribution mechanism for unwanted files, such as pornography or pirated software. The ability to write to the file system may be used to enable these attacks.

Countermeasure














Root should own all files in the FTP directory tree and the permissions should be set to 444. Executable files in the /bin directory should have the permissions set to 111. If you need to allow a user to upload files, the files should be set to be unreadable until they are reviewed. It is advisable that only one otherwise empty directory should be made writeable for so that users may upload files into it.

Figure 5.2: An extract from the Cisco Secure Scanner report

5.2.3.2 Cisco Secure Scanner vulnerability database

Of the 13 harmonised vulnerability categories, categories 3, 4, 7, 8, 10, 11, 12 and 13 are covered in very little detail, if at all, by the Cisco Secure Scanner's vulnerability database, as shown in table 5.3.

Table 5.3: Harmonised vulnerability categories covered by Cisco Secure Scanner

Harmonised vulnerability category	Cisco Secure Scanner
1  Password cracking and sniffing	✓
2  Network and system information gathering	✓
3  User enumeration and information gathering	✗
4  Backdoors, Trojans and remote controlling	✗
5  Unauthorised access to remote connections & services	✓
6  Privilege and user escalation	✓
7  Spoofing or masquerading	✗
8  Misconfigurations	✗
9  Denial-of-services (DoS) and buffer overflows	✓
10  Viruses and worms	✗
11  Hardware specific	✗
12  Software specific and updates	✗
13  Security policy violations	✗

5.2.4 SAINT

The Security Administrator's Integrated Network Tool (SAINT) [SAIN 03] is discussed because it was freely available until recently and supports the use of CVE. The SAINT can run on UNIX and LINUX operating systems and also scans for vulnerabilities on multiple operating systems. The SAINT is also available in an online version.

5.2.4.1 Practical experience with the SAINT

Because the SAINT incorporates CVE into its vulnerability database, standard vulnerability names are used. In addition, CVE's web site also has more information available on how to fix the detected vulnerabilities. This is a major advantage of the SAINT. The disadvantage of the SAINT is that it categorises its vulnerabilities into 177 categories, which makes it difficult to work with. It is better to have fewer vulnerability categories that are more manageable, as the harmonised vulnerability categories suggest.

5.2.4.2 SAINT vulnerability database














Of the 13 harmonised vulnerability categories, categories 1, 3, 4, 7, 10, 11 and 13 are covered in very little detail, if at all, by the SAINT's vulnerability database, as shown in table 5.4.

5.2.5 Internet Security Scanner (ISS)

The ISS version 6.2.1 is discussed because the ISS was one of the first VS products available on the software market. It is established and widely used in the industry today. There is an ISS version [ISSN 03] that can be downloaded from the Internet free of charge with full functionality, but it is limited to scan only the host on which it is installed.

The ISS supports the CVE standard to enable users to easily determine whether issues with different names are the same, and to allow for efficient sharing of security information. A CVE reference, however, may not exist for every vulnerability check used in the ISS and because of this CVE is only partially supported by the ISS.

Table 5.4: Harmonised vulnerability categories covered by SAINT

Harmonised vulnerability category	SAINT
1  Password cracking and sniffing	x
2  Network and system information gathering	✓
3  User enumeration and information gathering	x
4  Backdoors, Trojans and remote controlling	x
5  Unauthorised access to remote connections & services	✓
6  Privilege and user escalation	✓
7  Spoofing or masquerading	x
8  Misconfigurations	✓
9  Denial-of-services (DoS) and buffer overflows	✓
10  Viruses and worms	x
11  Hardware specific	x
12  Software specific and updates	✓
13  Security policy violations	x

5.2.5.1 Practical experience with the ISS

The ISS was installed on a Windows workstation and then set up to scan workstations and servers connected to the network for the vulnerabilities as specified in its vulnerability database. The ISS runs on Windows and has a very good user interface, but it can also scan for vulnerabilities on other operating systems such as UNIX. Depending on the size of the network and the specific scan policy that is set up before the scan can commence, the ISS scans the network for vulnerabilities and is relatively fast. A scan on a Windows workstation was completed in just over four minutes before a report was generated. Figure 5.3 shows an extract of one of the vulnerabilities in this report.

Modem detected and active (Active Modem)	
Risk Level:	Medium
Platforms:	Windows NT, Windows 95, Windows 98, Windows 2000, Windows ME
Description:	An active modem driver was detected. This situation only occurs when the modem is in use, or when the modem driver program is active. Modems can be a sign of an unauthorized channel around your firewall. Attackers could use a modem within the network to circumvent network security.
Remedy:	The modem must not be active while the computer is attached to the network. You may want to minimize the impact of a security breach caused by an unauthorized modem use by limiting which systems trust the computer using the modem. If using a modem on the network is required, configure all Remote Access Setup ports so that the port usage can dial-out only. Verify that your dial-out network configuration protocols match exactly the protocols you need to access the remote network. Review share permissions and account security to verify that the file system is not accessible from a remote location.
References:	ISS X-Force Modem detected and active http://xforce.iss.net/static/1292.php














Figure 5.3: An extract from the ISS report

The advantages of the ISS report are that it contains good and detailed descriptions and remedy procedures. In addition, a reference to additional information for the specific vulnerability detected is provided, as well as information on which operating system platforms the particular vulnerability can occur. Another major advantage is that the ISS classifies the particular vulnerability into a low-, medium-, or high-risk factor so that the rectification of vulnerabilities can be prioritised. The disadvantage of this report is that it requires effort to work through because of its large size.

5.2.5.2 ISS vulnerability database

Of the 13 harmonised vulnerability categories, categories 3, 6, 7, 8 and 10 are covered in very little detail, if at all, by the ISS's vulnerability database, as shown in table 5.5 below.

Table 5.5: Harmonised vulnerability categories covered by ISS

Harmonised vulnerability category	ISS
1  Password cracking and sniffing	✓
2  Network and system information gathering	✓
3  User enumeration and information gathering	✗
4  Backdoors, Trojans and remote controlling	✓
5  Unauthorised access to remote connections & services	✓
6  Privilege and user escalation	✗
7  Spoofing or masquerading	✗
8  Misconfigurations	✗
9  Denial-of-services (DoS) and buffer overflows	✓
10  Viruses and worms	✗
11  Hardware specific	✓
12  Software specific and updates	✓
13  Security policy violations	✓

5.2.6 Nessus Security Scanner

The Nessus Security Scanner is discussed because it is a widely known freeware product [TALI 00]. The Nessus Security Scanner executes mainly on UNIX-based platforms, but it can scan for vulnerabilities on multiple operating system platforms. The Nessus Security Scanner is built upon client-server architecture where the server works on a UNIX-based platform. Different clients are available that can run on a UNIX or Windows operating system platform. The Nessus Security Scanner also supports CVE references.

5.2.6.1 Practical experience with the Nessus Security Scanner

The Nessus Security Scanner works on the concept of plug-in architecture. This means that there is a plug-in for each vulnerability that the Nessus Security Scanner can check for. This way, it is easy to add new vulnerability signatures as external plug-ins when they become available. These can simply be downloaded from the Nessus Security Scanner web site [DERA 03] via FTP.














It is also possible to add customised vulnerability signatures . To be able to do this, the Nessus Security Scanner includes the Nessus Attack Scripting Language (NASL), which is a language designed to write customised vulnerability signatures easily and quickly. These plug-ins then also constitute the vulnerability database for the Nessus Security Scanner.

The main advantage of the Nessus Security Scanner is that it is very fast. The vulnerability tests performed by the Nessus Security Scanner co-operate so that nothing is done that is not necessary . For example, if an FTP server is found not to offer anonymous logins, then anonymous-related vulnerability checks will not be attempted or performed for anonymous FTP vulnerabilities, which saves time . Some VS products will attempt to scan for anonymous FTP vulnerabilities, if their scan policies were set up to do that, even if no anonymous FTP vulnerabilities are present. This causes those VS products to waste valuable time since they will not continue to scan for the next vulnerability, as defined by their scan policy, until scanning for anonymous FTP vulnerabilities has timed out. Another advantage of the Nessus Security Scanner is that it categorises the risk level of each vulnerability from low to very high in the report that it generates, enabling the prioritisation of the urgency of fixing the vulnerabilities found. The disadvantage of this report, however, is that it requires effort to work through because of its large size.

5.2.6.2 Nessus Security Scanner vulnerability database

Of the 13 harmonised vulnerability categories, categories 1, 3, 7, 8, 10, 11 and 13 are covered in very little detail, if at all, by the Nessus Security Scanner's vulnerability database, as shown in table 5.6.

Table 5.6: Harmonised vulnerability categories covered by Nessus Security Scanner

Harmonised vulnerability category	Nessus Security Scanner
1  Password cracking and sniffing	x
2  Network and system information gathering	✓
3  User enumeration and information gathering	x
4  Backdoors, Trojans and remote controlling	✓
5  Unauthorised access to remote connections & services	✓
6  Privilege and user escalation	✓
7  Spoofing or masquerading	x
8  Misconfigurations	x
9  Denial-of-services (DoS) and buffer overflows	✓
10  Viruses and worms	x
11  Hardware specific	x
12  Software specific and updates	✓
13  Security policy violations	x

5.3 SUMMARY OF CURRENT VS PRODUCTS

In the previous sections different VS products were discussed. In essence, all these products have one main goal: identifying vulnerabilities. However, the way in which these VS products go about accomplishing this goal often differs extensively from one VS product to another. In addition, these different VS products do not all scan for exactly the same types of vulnerabilities. Fortunately, by making use of harmonised vulnerability categories [VEE2 03], a measure is available to identify how the different VS products comply with harmonised vulnerability categories.

Figure 5.4 shows a mapping, compiled during this research project, of the vulnerabilities found for each of the five VS products discussed in the previous sections onto the harmonised vulnerability categories. The mapping process was done for each individual VS product. The vulnerability database of a specific VS product was carefully dissected by studying each vulnerability as defined in the vulnerability

database. A particular vulnerability was then allocated to one of the 13 harmonised vulnerability categories.

5.3.1 Mapping onto harmonised vulnerability categories

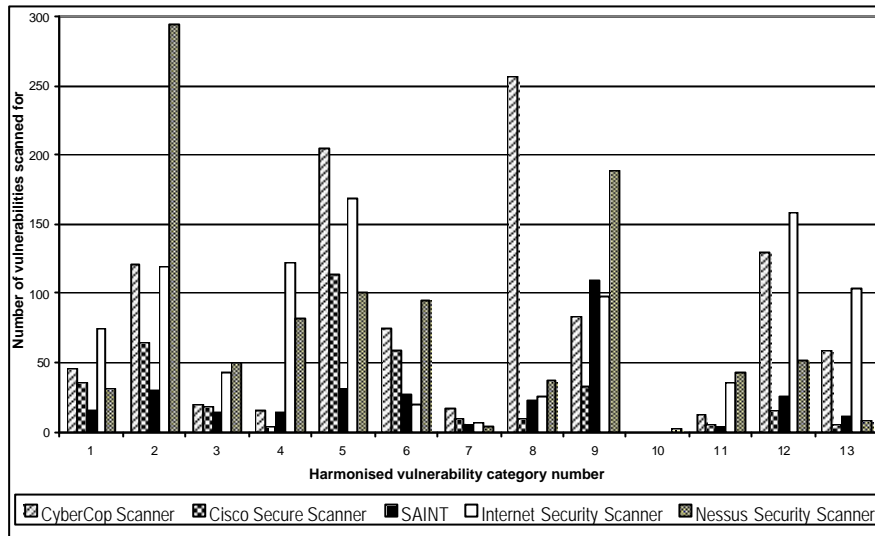


Figure 5.4: Vulnerability mapping of different VS products onto the harmonised vulnerability categories

From figure 5.4 it is clear that the different VS products comply differently with the 13 harmonised vulnerability categories. For example, the Nessus Security Scanner can detect far more *network and system information gathering* (category 2) vulnerabilities than all the other VS products. The CyberCop Scanner, on the other hand, outperforms all the other VS products when detecting *misconfiguration* (category 8) vulnerabilities. In addition, only one VS product, namely the Nessus Security Scanner, scans for *viruses and worms* (category 10) and only for a very limited number of viruses and worms. In almost all the harmonised vulnerability categories, the ISS scans for more vulnerabilities than the other VS products. The ISS, therefore, seems to be the VS product with the highest number of vulnerabilities that it can scan for across the harmonised vulnerability categories.

In figure 5.4, significant differences can be noticed in some harmonised vulnerability categories between the number of vulnerabilities that can be scanned for by the

different VS products. The following section will elaborate on and discuss the significance of these differences.

5.3.2 Differences in VS products

More might be read into the data displayed in figure 5.4. The harmonised vulnerability categories 2, 4, 8, 9 and 13, as shown in figure 5.4, will be discussed in more detail to examine why there are such major differences in the number of vulnerabilities that each of the VS products can scan for. These five harmonised vulnerability categories were specifically chosen because there is a considerable difference in the number of vulnerabilities that can be scanned for by the particular VS that is able to scan for the highest number of vulnerabilities, and the VS that is able to scan for the second highest number of vulnerabilities for each specific category.

The sections that follow will briefly look at these differences and discuss the significance of each.

5.3.2.1 **i** 2: Network and system information gathering

An extract from figure 5.4 of harmonised vulnerability category 2, *network and system information gathering*, is shown in figure 5.5.

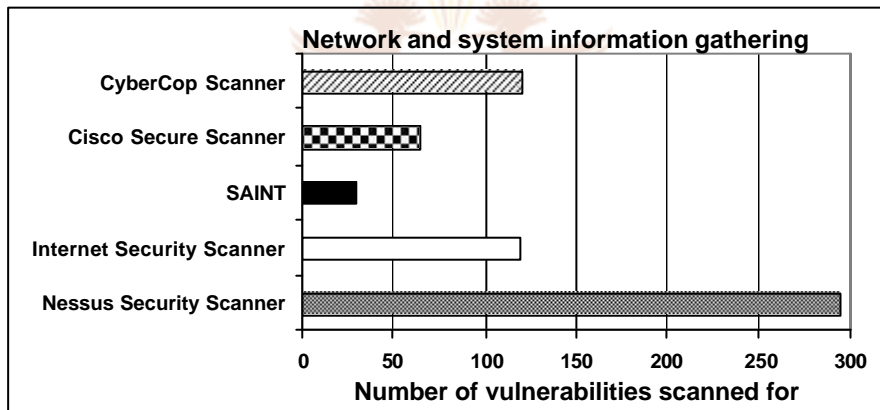


Figure 5.5: Number of vulnerabilities scanned for by different VS products for harmonised vulnerability category 2: network and system information gathering

The Nessus Security Scanner scans for the highest number of *network and system information gathering* vulnerabilities (294), while the ISS scans for the second highest

(119) in this harmonised vulnerability category. To ascertain whether this difference is really that significant, examples of the most important *network and system information gathering* vulnerabilities for each of these two VS products are given in table 5.7.

Table 5.7: Important network and system information gathering vulnerabilities

Nessus Security Scanner	ISS
Gathering information about the common gateway interface (CGI) of a Web server	Gathering information about the users registered on a system
Gathering information about remote procedure call (RPC) services	Gathering information about different services installed on a system
Gathering information about the file transfer protocol (FTP) service	Gathering information about the physical route that can be traced to a system

Gathering information about users as performed by the ISS is perhaps a more important vulnerability than the gathering of CGI information by the Nessus Security Scanner. Gathering information about users should therefore be given higher priority. As clearly shown in figure 5.5, the ISS detects far fewer *network and system information gathering* vulnerabilities than the Nessus Security Scanner. The Nessus Security Scanner scans for more vulnerabilities than the ISS over all the harmonised vulnerability categories in total. In this case the major difference in the number of *network and system information gathering* vulnerabilities that these two VS products are able to detect is **not** significant.

5.3.2.2 4: Backdoors, Trojans and remote controlling

An extract from figure 5.4 of harmonised vulnerability category 4, *backdoors, Trojans and remote controlling*, is shown in figure 5.6.

The ISS scans for the highest number of *backdoors, Trojans and remote controlling* vulnerabilities (122), while the Nessus Security Scanner scans for the second highest (78) in this harmonised vulnerability category. To ascertain whether this difference is significant, examples of the most important *backdoors, Trojans and remote controlling* vulnerabilities for each of these two VS products are given in table 5.8.

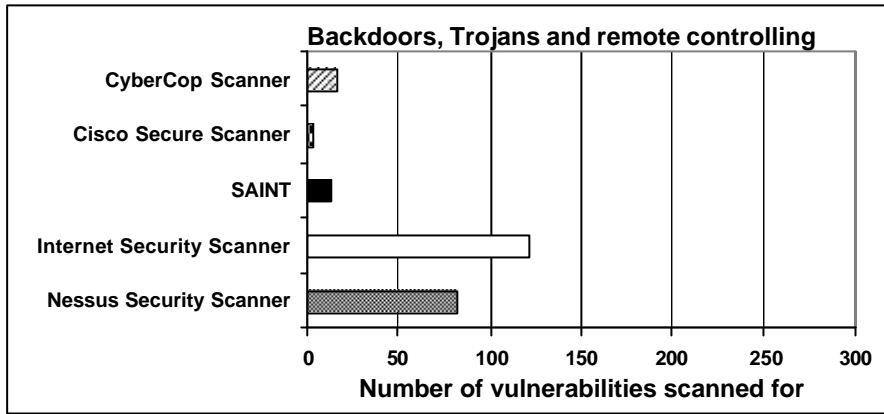


Figure 5.6: Number of vulnerabilities scanned for by different VS products for harmonised vulnerability category 4: backdoors, Trojans, and remote controlling

Table 5.8: Important backdoors, Trojans, and remote controlling vulnerabilities

ISS	Nessus Security Scanner
Back Orifice backdoor found	Back Orifice backdoor found
Netbus backdoor found	Netbus backdoor found
Windows NT remote access service (RAS) enabled	PC Anywhere remote administration tool found

Both the ISS and the Nessus Security Scanner are able to detect more or less the same important *backdoors, Trojans, and remote controlling* vulnerabilities. Figure 5.6, however, shows that the Nessus Security Scanner detects fewer *backdoors, Trojans, and remote controlling* vulnerabilities than the ISS. In this case the difference in the number of *backdoors, Trojans, and remote controlling* vulnerabilities that these two VS products are able to detect is **definitely** significant, with the ISS being the best. The difference in the number of vulnerabilities is very large.

5.3.2.3 8: Misconfigurations

An extract from figure 5.4 of harmonised vulnerability category 4, *misconfigurations*, is shown in figure 5.7.

The CyberCop Scanner scans for the highest number of *misconfiguration* vulnerabilities (255), while the Nessus Security Scanner scans for the second highest (41) in this harmonised vulnerability category. To ascertain whether this difference is significant, examples of the most important *misconfiguration* vulnerabilities for each of these two VS products are given in table 5.9.

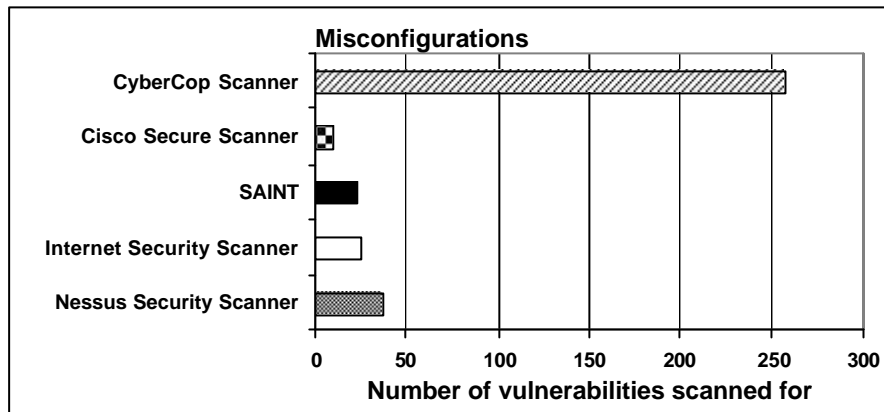


Figure 5.7: Number of vulnerabilities scanned for by different VS products for harmonised vulnerability category 8: misconfigurations

Table 5.9: Important misconfiguration vulnerabilities

CyberCop Scanner	Nessus Security Scanner
Default passwords, usernames and/or settings were found for different applications	Default passwords, usernames and/or settings were found for different applications
Internet Control Message Protocol (ICMP) enabled	Some ICMP settings enabled
NetBIOS shares found on the system with world-readable permissions found	SMB shares found on the system with world-readable permissions found

Both the CyberCop Scanner and the Nessus Security Scanner are able to detect more or less the same important *misconfiguration* vulnerabilities. As clearly shown in figure 5.7, however, the Nessus Security Scanner detects far fewer *misconfiguration* vulnerabilities than the CyberCop Scanner. The big difference in the number of *misconfiguration* vulnerabilities that these two VS products can detect, is attributed to the fact that the entire vulnerability database of the CyberCop Scanner contain so much more vulnerability signatures than that of the Nessus Security Scanner. In this case the major difference in the number of *misconfiguration* vulnerabilities that these two VS products are able to detect is **definitely** significant and favours the CyberCop Scanner.

5.3.2.4 9: Denial-of-services (DoS) and buffer overflows

An extract from figure 5.4 of harmonised vulnerability category 4, *denial-of-services (DoS) and buffer overflows*, is shown in figure 5.8.

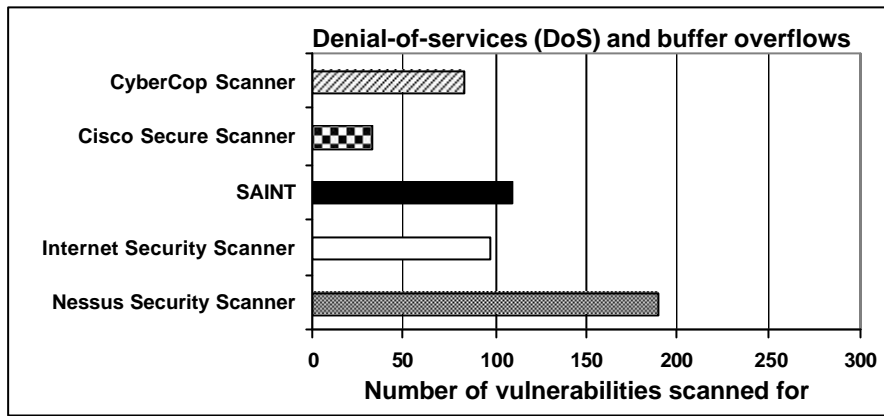


Figure 5.8: Number of vulnerabilities scanned for by different VS products for harmonised vulnerability category 9: denial-of-services (DoS) and buffer overflows

The Nessus Security Scanner scans for the highest number of *denial-of-service (DoS) and buffer overflow* vulnerabilities (192), while the SAINT scans for the second highest (110) in this harmonised vulnerability category. To ascertain whether this difference is significant, examples of the most important *denial-of-service (DoS) and buffer overflow* vulnerabilities for each of these two VS products are given in table 5.10.

Table 5.10: Important denial-of-service (DoS) and buffer overflow vulnerabilities

Nessus Security Scanner	SAINT
Microsoft Internet Information Server (IIS) and other HTTP-based DoS vulnerabilities found	Different hardware application buffer overflow vulnerabilities found
Berkley Internet Name Domain (BIND) and domain name service (DNS) DoS vulnerabilities found	DNS DoS vulnerabilities found
Different Web service DoS vulnerabilities found	Different database application and SQL buffer overflow vulnerabilities found

The Nessus Security Scanner can detect Microsoft IIS and BIND DoS vulnerabilities, which have more serious consequences than the hardware buffer overflow vulnerabilities detected by the SAINT. Detecting Microsoft IIS and BIND DoS vulnerabilities should therefore be given a higher priority. Figure 5.8 clearly shows that the SAINT detects far fewer *denial-of-service (DoS) and buffer overflow* vulnerabilities than the Nessus Security Scanner. The SAINT’s vulnerability database is almost three times smaller than that of the Nessus Security Scanner in terms of the total number of vulnerabilities it can detect over all harmonised vulnerability

categories. In this case the difference in the number of *denial-of-service (DoS) and buffer overflow* vulnerabilities that these two VS products are able to detect is **definitely** significant, with the Nessus Security Scanner being the best. It should also be mentioned that because the SAINT's vulnerability database is significantly smaller than that of the Nessus Security Scanner, it can be argued that the Nessus Security Scanner detects more *denial-of-service (DoS) and buffer overflow* vulnerabilities that are **not** as important, in the researcher's opinion.

5.3.2.5 13: Security policy violations

An extract from figure 5.4 of harmonised vulnerability category 4, *security policy violations*, is shown in figure 5.9.

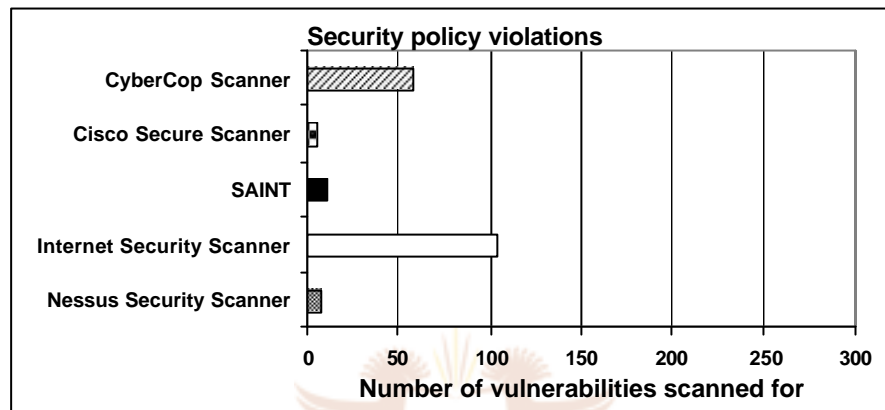


Figure 5.9: Number of vulnerabilities scanned for by different VS products for harmonised vulnerability category 13: security policy violations

The ISS scans for the highest number of *security policy violations* vulnerabilities (104), while the CyberCop Scanner scans for the second highest (59) in this harmonised vulnerability category. To ascertain whether this difference is significant, examples of the most important *security policy violations* vulnerabilities for each of these two VS products are given in table 5.11 below.

Table 5.11: Important security policy violations vulnerabilities

ISS	CyberCop Scanner
Password policy not sufficient	Password policy not sufficient
System auditing policy not set up	System event log or auditing policy not set up
Hardware access policy too lenient	Account access policy too lenient

Both the ISS and the CyberCop Scanner are able to detect more or less the same important *security policy violations* vulnerabilities. As shown in figure 5.9, the CyberCop Scanner detects fewer *security policy violations* vulnerabilities than the ISS. In this case the difference in the number of *security policy violations* vulnerabilities that these two VS products are able to detect is **definitely** significant, with the ISS performing the best.

5.4 CONCLUSION

This chapter discussed different VS products and looked at how each product differs in the way that it can scan for vulnerabilities. A useful means of dealing with the different ways in which vulnerabilities are scanned for is to find a common way of referring to vulnerabilities amongst different VS products. This can be accomplished by using CVE. CVE, however, still does not solve the problem of knowing which vulnerabilities different VS products scan for, because CVE does not categorise vulnerabilities. This problem can be solved by using harmonised vulnerability categories.

A mapping from a specific VS product's vulnerability database onto the harmonised vulnerability categories is a process that needs to be carried out for each VS product considered for implementation by an organisation. Harmonised vulnerability categories prove to be a supporting mechanism for reviewing different VS products to determine how a specific VS product addresses the scope of vulnerabilities as defined by the harmonised vulnerability categories.

VS products can differ extensively from each other in terms of the number of vulnerabilities that each VS is able to detect. This is mainly due to the fact that some VS products employ a vulnerability database containing many vulnerability signatures while other VS products employ a small vulnerability database. Although a specific VS product may contain a large vulnerability database, however, many of its vulnerability signatures may be outdated or not so important. The importance factor of vulnerabilities in the harmonised vulnerability categories is addressed in the current research project by priority levels, which will be discussed in the next chapter.



CHAPTER 6

VULNERABILITY FORECASTING – A CONCEPTUAL MODEL

6.1 INTRODUCTION

The previous chapters discussed different state-of-the-art information security technologies that can be used to secure computer systems and networks, such as intrusion detection systems (IDSs) and vulnerability scanners (VSs). These specific information security technologies were discussed because they have contributed significantly to the field of information security in recent times and they are the latest developments in information security. It was VSs, however, that attracted the attention of the researcher because they follow a **proactive** approach to finding and minimising vulnerabilities, whereas IDSs follow a **reactive** approach.

The proactive approach to finding and minimising vulnerabilities is considered to be a better approach, because it is based on the principle of prevention being better than cure. Although the proactive behaviour of VSs is a positive point, there are still many problems with state-of-the-art VSs. This chapter will identify these problems and suggest which of them will be addressed in this research. A conceptual model is then introduced that will address some of these problems and a functional discussion of what the conceptual model is trying to achieve is given.

6.2 PROBLEMS WITH STATE-OF-THE-ART VSS

Despite their many shortcomings, VSs have proven successful in combating most vulnerabilities. One of their biggest drawbacks, however, is the fact that they have to “recognise” a vulnerability before they can detect it, and for a VS to “recognise” a vulnerability, it must have access to a list featuring the “signature” of the vulnerability in question. This list is commonly referred to as a vulnerability database. If a

completely new vulnerability is identified, the vulnerability database has to be updated with the signature of the said new vulnerability. After adding the signature of the new vulnerability to the vulnerability database, the network needs to be scanned again to ensure that it does not contain the newly identified vulnerability, especially since new vulnerabilities appear like clockwork. For this reason, the network of an organisation needs to be scanned on a daily basis – failing which, its VS would be rendered obsolete.

When conducting a scan, a VS generally occupies a vast number of network and system resources. For this reason, a scanning exercise not only becomes too costly to undertake every day, but also too time-consuming, especially in view of the fact that a single scan conducted on a relatively small network could last for hours. In this way, the network utilisation may, on occasion, come to an abrupt halt when checking for denial-of-service vulnerabilities. To make matters worse, it is considered critical for distributed applications, such as online reservation systems, to utilise all of their available network bandwidth, as insufficient bandwidth could cause such applications to fail. In addition, when scanning for password vulnerabilities, the processing ability of a system may be impeded to the extent of compromising the processing capacity required for mission-critical tasks.

VSs also lack intelligence [SCHN 00] in the sense that they are unable to automatically identify new vulnerabilities and automatically update the vulnerability database accordingly. In addition, specialised skills are required to interpret and productively apply the results of a scan conducted by a VS.

The model and structure of specific VS products differ extensively. For example, Nessus Security Scanner [DERA 03] is a VS product that includes the Nessus Attack Scripting Language (NASL), which is a language designed to write customised vulnerability signatures easily and quickly. Each such signature is then added as a plug-in to Nessus Security Scanner. These plug-ins also comprise the vulnerability database for Nessus Security Scanner. This is in contrast to other VS products, for example Internet Security Scanner (ISS) [ISSN 03], which has a conventional database, i.e. in the Microsoft Access Database format, in which its vulnerabilities are

stored. As new vulnerabilities emerge, this conventional database is updated or replaced rather than new signature plug-ins simply being added as they become available, as in the case of Nessus Security Scanner. This poses compatibility problems when vulnerabilities from one VS product need to be compared to those of another.

In theory, scans should be conducted at regular intervals. In practice, however, this is not always possible. Scans may be conducted at irregular intervals when, for example, a scheduled scan has to be postponed or even abandoned in favour of a mission-critical event such as an unscheduled backup.

There will always be an administrative overhead in updating the vulnerability database of a specific VS product at regular intervals, the best time being before a scan is conducted. This practice is not always followed, however, and may result in new vulnerabilities not being detected when a vulnerability scan is conducted.

The rectification procedures provided in a scan report may be such that attempting to automate the rectification procedure is currently unfeasible. Skilled human resources, therefore, are still needed to do the job.

Since the rectification procedures, which most VS products offer on the vulnerabilities they might uncover, are left entirely to skilled human resources, no automated high-level risk management procedures are suggested by VS products. However, the scan report that a VS product generates can be seen as some kind of low-level risk management since, after a scan report has been created, it provides some rectification procedures providing human resources with specific steps on how to fix the vulnerabilities that were uncovered during the scan. The high-level risk management required for this purpose, thus, refers to a higher level that incorporates the combined use of harmonised vulnerability categories and intelligent techniques in a bid to provide information that would enable an organisation to proactively act on vulnerabilities.

In summary, table 6.1 lists the problems identified with state-of-the-art VS products. The current research project will, however, be dedicated to the improvement of

existing VS products and VS technology by concentrating on specific problems in particular, as indicated by table 6.1.

Table 6.1 : Problems identified and addressed regarding state-of-the-art VS products

Problems identified	Problems addressed
1. Conducting vulnerability scans is too time-consuming.	✓
2. A VS product generally occupies a vast number of network and system resources, leading to the degradation of system performance while vulnerability scans are being conducted.	✓
3. VS products lack intelligence because they are unable to learn about new vulnerabilities automatically.	✓
4. The vulnerability database structure differs extensively from one VS product to another.	✓
5. The types of vulnerabilities being scanned for differ extensively from one VS product to another.	✓
6. Scans may not always be conducted at regular intervals due to unforeseen circumstances, for example when critical maintenance on servers and the network is carried out.	✓
7. The vulnerability database should be updated before a scan is conducted, otherwise the scan result may not be accurate enough.	
8. Most rectification procedures cannot be automated and still require the expertise of qualified personnel.	
9. VS products do not provide adequate and sufficient information that would aid high-level risk management.	✓

In order to minimise the impact of the problems outlined above, the researcher would like to introduce the concept of **vulnerability forecasting**.

6.3 CONCEPT OF VULNERABILITY FORECASTING

6.3.1 Defining the term “vulnerability forecasting”

The term “vulnerability forecasting” (VF) can be defined as “that attempt to identify potential vulnerable areas on hosts across a network and to what extent such areas on hosts across a network will be vulnerable over a specific period in the near future”. The principal aim of VF is, therefore, to predict trends or patterns in which potential vulnerabilities could occur. Knowing what such a vulnerability forecast is means that proactive action can be taken in a bid to minimise the risks that such vulnerabilities may pose.

6.3.2 A conceptual model for VF

The conceptual VF model will be presented over two levels: level 1, which is a high-level design, and level 2, which is a low-level design of the conceptual model.

6.3.2.1 Level 1 of the conceptual VF model

The high-level design of the conceptual VF model comprises three main components and is depicted in figure 6.1 below:

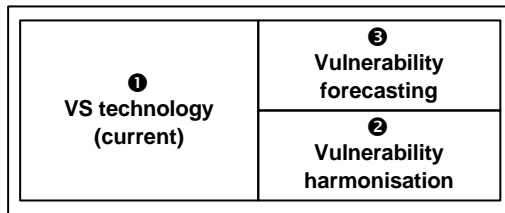


Figure 6.1: Level 1 – A conceptual model for vulnerability forecasting

A brief description of the main components in figure 6.1 follows:

1. VS technology (current)

- This component constitutes one or more state-of-the-art VS products that are used for collecting the data needed for VF.

2. Vulnerability harmonisation

- This component serves as a coupler between the VS technology and the vulnerability forecasting components in a bid to “standardise” the VS product’s output into a harmonised form.

3. Vulnerability forecasting

- This component does the actual intelligent vulnerability forecast.

6.3.2.2 Level 2 of the conceptual VF model

Each of the main components of the conceptual VF model, as introduced in the previous section, contains subcomponents. As an integral part of the VF model, most of these subcomponents include a variety of databases. These databases are all integrated into the VF model and will be referred as the **VF logical database** for the purposes of this model. This logical database is built up as the chapter progresses by systematically adding components to it whilst discussing the components. The subcomponents are discussed in the sections that follow on the second level – a more detailed level – of the conceptual VF model.

6.3.2.2.1 VS technology (current)

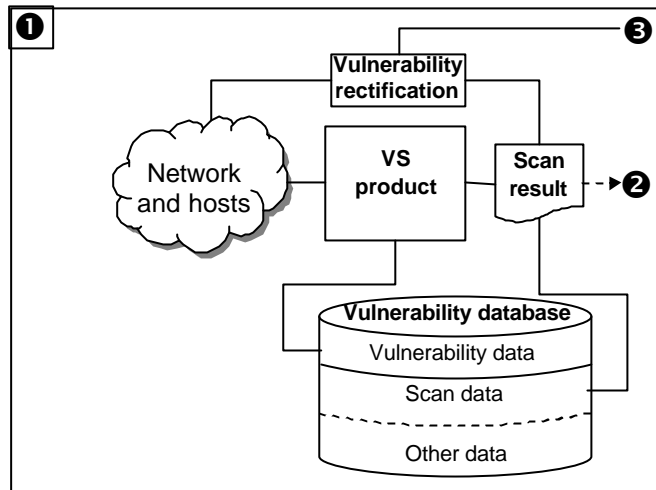


Figure 6.2: The VS technology (current) component

Figure 6.2 represents the VS technology (current) component of the conceptual VF model introduced in the previous section. The reason for using current VS technology in the VF model enables the use of existing technology rather than attempting to design yet another module in the VF model. In addition, any current VS product can be used in the conceptual VF model, rendering the conceptual VF model more flexible. None of the subcomponents in figure 6.2 are, therefore, revolutionary or new to the conceptual VF model, but they are state-of-the-art technology used as an integral part of the proposed conceptual VF model. In summary, a VS product analyses the security state of a network of hosts on the basis of information collected, referred to as scans, at different intervals. After a scan is completed, the VS product generates scan results in the form of a report that states all the vulnerabilities found during the scan and leaves it up to a person to rectify these vulnerabilities.

Network and hosts

The *network and hosts* component constitutes all computer systems interconnected in a network or subnetwork. This can also refer to multiple networks spread across the Internet. The hosts interconnected to the network constitute not only personal computers and servers, but may also include hardware devices, for example routers, switches, hubs and network printers. All of these systems may contain vulnerabilities in some form, which can be detected by VS products.

VS product

The *VS product* component constitutes a state-of-the-art VS product that is used to conduct vulnerability scans on hosts across a network. It is also possible to employ more than one VS product to get more accurate results, because not all VS products scan for exactly the same vulnerabilities. In other words, employing more than one VS product may improve the chances of identifying more vulnerabilities.

The vulnerability database actually forms part of a VS product. However, this component will be discussed separately from the VS product since the vulnerability database is an important component in the conceptual VF model.

Vulnerability database

The *vulnerability database* is a database linked to the VS product. The vulnerability database model and structure can differ considerably from one VS product to another. Different VS products were therefore evaluated whilst conducting this research in a bid to reproduce a generalised view of the vulnerability database as implemented by many state-of-the-art VS products.

The vulnerability database is subdivided into two main parts: vulnerability data and scan data. **Vulnerability data** constitutes the first part of the VF logical database and includes the signatures of all known weaknesses in software or hardware as found in the network and hosts connected to the network. The vulnerability data can be generalised into an entity-relationship diagram as depicted in figure 6.3. The relationship in figure 6.3 is represented as “one-to-many” – also denoted as 1 to M.

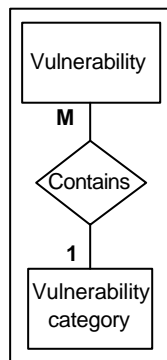


Figure 6.3: VF logical database part 1: vulnerability data

The vulnerability data of the VF logical database consists of the following two entities:

- **Vulnerability category**
 - This entity contains the specific categories into which vulnerabilities are classified by the specific VS product. This classification of vulnerabilities is done by the VS product vendor. The minimum fields in this entity include a *category number* and a *category description*.
- **Vulnerability**
 - This entity contains each specific vulnerability signature that is checked for when a scan is conducted by the specific VS product. The minimum fields in this entity include a *vulnerability number*, a *vulnerability description*, a *vulnerability priority*, a *rectification procedure* recommended for the specific vulnerability, a common vulnerabilities and exposures (CVE) number and a *category number* used as the link to the vulnerability category entity.

Figure 6.4 represents a typical vulnerability data report generated from the vulnerability data by a VS product.

Vulnerability #	Vulnerability description	Vulnerability category #	Vulnerability category description
1	Anonymous FTP enabled	1	FTP vulnerabilities
2	FTP root directory write-enabled	1	
3	Password file not shadowed	2	Password vulnerabilities
4	Unpassworded Laser Jet printer	2	
5	IRC server present	3	Information gathering vulnerabilities
6	Can trace route to host	3	
7	ICMP backdoor found	4	Backdoor vulnerabilities
8	Sendmail syslog buffer overflow	5	E-mail vulnerabilities
9	Mail forgery	5	
10	Sendmail relaying allowed	5	
.....

Figure 6.4: Vulnerability data report

Scan data constitutes the second part of the VF logical database. It contains data such as the specific vulnerability scan ID, the specific host on which the scan was conducted and the detailed result of each scan. The scan data can be generalised into an entity-relationship diagram as depicted in figure 6.5.

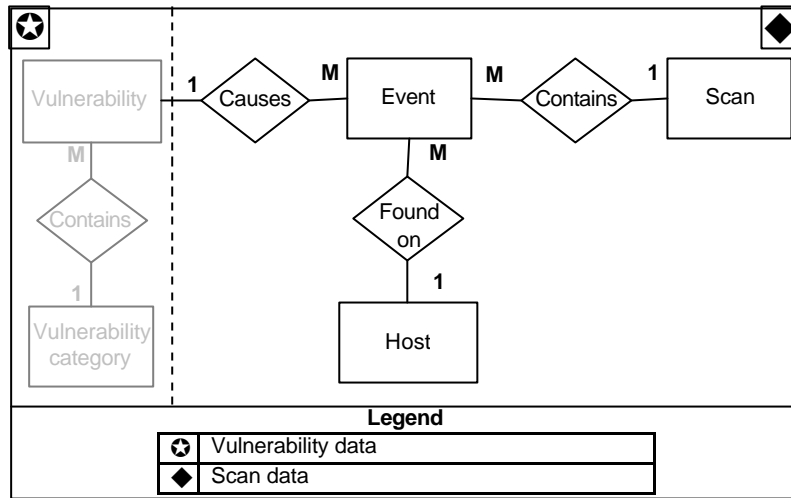


Figure 6.5: VF logical database part 2 added: scan data

The scan data of the VF logical database consists of the following three entities:

- **Event**
 - This entity contains each specific vulnerability that was found by the specific VS product whilst conducting a scan. The minimum fields in this entity include an *event number*, a *vulnerability number*, a *date & time* when the vulnerability was found, a *host number* used as the link to the host entity in order to know where the vulnerability was found, a *scan number* in order to link the specific event to a specific vulnerability scan that was conducted and a *harmonised vulnerability category number* in order to link the specific event to a harmonised vulnerability category, which will be discussed later in the chapter. The event entity refers only to the data of the very last scan conducted. Before a new scan is conducted, thus, the content of the event entity is first archived to a history entity, after which the event entity is cleared

in order to store the data of the new scan to be conducted. The history entity will be discussed later in this chapter.

- **Host**
 - This entity contains each specific host that is scanned by the VS product in the network. The minimum fields in this entity include a *host number*, i.e. the host's IP address, and a *host description*.
- **Scan**
 - This entity contains each specific scan that was conducted by the VS product. The minimum fields in this entity include a *scan number*, and a *date* in order to know when a specific scan was conducted.

Figure 6.6 represents a typical scan data report generated for one specific vulnerability scan that was conducted by a VS product.

Event #	Date & time	Vulnerability #	Host #	Scan #
1	2003-04-15, 10:55:23	1	192.168.1.2	1
2	2003-04-15, 10:55:42	3		
3	2003-04-15, 11:01:13	51	192.168.1.3	
4	2003-04-15, 11:13:21	21		
5	2003-04-15, 11:13:25	42	192.168.1.7	
6	2003-04-15, 11:42:47	4		
7	2003-04-15, 12:58:51	5		
8	2003-04-15, 12:59:58	38	192.168.1.8	
9	2003-04-15, 13:00:14	71		
10	2003-04-15, 13:06:38	23		
...	

Figure 6.6: Scan data report for scan 1

Scan result

After a scan has been conducted, the scan result report is generated by the VS product containing the *scan result* for one vulnerability scan. The scan result report contains all the information as shown in the scan data report in figure 6.6, but some additional fields are given as well. This report is normally archived by the VS product and stored in the vulnerability database. The scan result report usually contains the following information for each vulnerability found:

- The event number that uniquely identifies each event that occurred each time a vulnerability was found on a particular host.
- The host number – typically an IP address – on which the vulnerability was found.
- The date and time when the vulnerability was found.
- The vulnerability number and description of the specific vulnerability found.
- The rectification priority for the specific vulnerability found.
- The appropriate CVE number for the specific vulnerability found.

An example of a scan result report is shown in figure 6.7. Each vulnerability that is scanned for causes an event to occur, which is recorded in the scan result report. As soon as all vulnerabilities have been scanned for on one particular host, the VS product moves on to the next host until all hosts in the specific network have been scanned. Each time a vulnerability is found, it is added to the scan result report for the particular scan. Note that most VS products are able to scan various hosts in parallel and not necessarily in a specific order. However, for the sake of simplicity, figure 6.7 shows the scan result report as if hosts were scanned one-by-one in host order.

Vulnerability rectification

The vulnerabilities that were detected by a vulnerability scan procedure need to be rectified. The *vulnerability rectification* procedure is a manual process, in other words it requires skilled human resources to work through the generated vulnerability report to rectify the vulnerabilities. The fact that this is a manual process is a concern for vulnerability scanning, since it may take days for human resources to manually rectify vulnerabilities.

Event #	Host #	Date & time	Vulnerability #	Vulnerability description	Priority	Rectification procedure	CVE	
1	192.168.1.2	2003-04-15, 10:55:23	1	Anonymous FTP enabled	M	Go to the FTP settings of the FTP server. Disable anonymous FTP. Set the "shadow" attribute of the password file to "true".	19990456	
2		2003-04-15, 10:55:42	3	Password file not shadowed	H		20021024	
...	
6		2003-04-15, 11:42:47	4	FTP root directory write-enabled	L		19982300	
7		2003-04-15, 12:58:51	2	IRC server present	L		20023410	
8		2003-04-15, 12:59:58	3	Password file not shadowed	H		20021024	
...	192.168.1.7	
14		2003-04-14, 13:09:45	4	FTP root directory write-enabled	L	19982300		
...		
56	192.168.1.41	2003-04-14, 15:41:12	1	Anonymous FTP enabled	M	Go to the FTP settings of the FTP server. Disable anonymous FTP. Stop/uninstall the IRC server. See http://www.abcd.com for more info.	19990456	
57		2003-04-14, 15:48:00	2	IRC server present	L		20023410	
58		2003-04-14, 15:56:04	5	Can trace route to host	L		20000343	
...	

Figure 6.7: The scan result report: vulnerabilities found on different hosts on a network during a specific scan

6.3.2.2.2 Vulnerability harmonisation

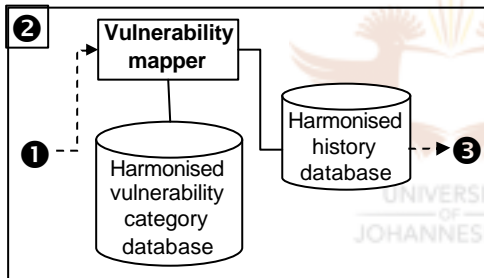


Figure 6.8: The vulnerability harmonisation component













Figure 6.8 represents the vulnerability harmonisation component of the conceptual VF model. This component does not do the actual vulnerability forecasting yet, but serves as an in-between process where the data it received from component 1 of the conceptual VF model is transformed in such a way that it is “harmonised” and, thus, prepared to be “understood” by component 3 of the conceptual VF model. In summary, the output of the VS product in component 1 of the conceptual VF model as

shown in figure 6.1, namely the scan result, serves as input to the vulnerability mapper in component 2 of the conceptual VF model. The vulnerability mapper maps the vulnerabilities found by the VS product onto the harmonised vulnerability categories and stores the result in the harmonised history database. This process is repeated each time a vulnerability scan is conducted.

Harmonised vulnerability category database

The *harmonised vulnerability categories* are rather static. They are stored in a database which is not updated unless a new breed of vulnerabilities evolves, leading to the creation or modification of an additional harmonised vulnerability category. The harmonised vulnerability categories are shown in table 6.2 below and are discussed in detail in chapter 4.

Table 6.2 : Summary of the harmonised vulnerability categories

Harmonised vulnerability category number, icon, and name	Harmonised vulnerability category description
1  Password cracking and sniffing	Vulnerabilities with a root cause of having accounts with weak or no passwords
2  Network and system information gathering	Vulnerabilities concerned with scanning a network to discover a map of available hosts and vulnerable services
3  User enumeration and information gathering	Vulnerabilities concerned with retrieving information of user accounts on a specific system
4  Backdoors, Trojans and remote controlling	Vulnerabilities concerned with having hidden access mechanisms installed on a system
5  Unauthorised access to remote connections & services	Vulnerabilities concerned with the risk that an unauthorised person has the ability to connect to and misuse a system
6  Privilege and user escalation	Vulnerabilities concerned with the risk that the access rights of an existing user account can be upgraded by an unauthorised user, granting more privileges to the user
7  Spoofing or masquerading	Vulnerabilities concerned with the risk that an intruder can fake an IP address in a bid to act as another person
8  Misconfigurations	Vulnerabilities concerned with the risk that applications have been incorrectly configured
9  Denial-of-services (DoS) and buffer overflows	Vulnerabilities concerned with the risk of one or more intruders launching an attack designed to disrupt or deny legitimate users' or applications' ability to access resources
10  Viruses and worms	Vulnerabilities concerned with malicious programs
11  Hardware specific	Vulnerabilities concerned with having hardware peripherals that execute ROM-based or firmware-based programs
12  Software specific and updates	Vulnerabilities concerned with the risk that specific applications contain specific, well-known bugs

13	Security policy violations	Vulnerabilities concerned with the risk that an Internet security policy has been violated
----	----------------------------	--

The harmonised vulnerability category database contains the *harmonised vulnerability category* entity, as shown in figure 6.9, in context with the generalised vulnerability database. There is only one entity in the harmonised vulnerability category database:

- **Harmonised vulnerability category**
 - This entity contains the harmonised vulnerability categories, as defined in this research, into which all current vulnerabilities are classified. This classification of vulnerabilities was done by the researcher. The minimum fields in this entity include a *harmonised vulnerability category number*, a *harmonised vulnerability category icon*, a *harmonised vulnerability category name*, and a *harmonised vulnerability category description*.

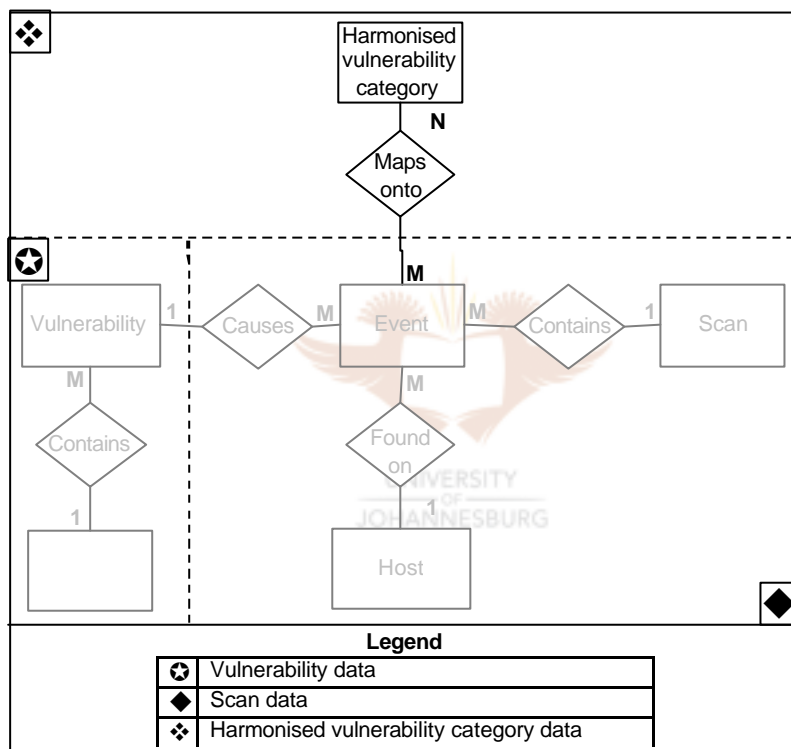


Figure 6.9: VF logical database part 3 added: *harmonised vulnerability category data*

Vulnerability mapper

The *vulnerability mapper* serves as a “translator” in the sense that, for the vulnerabilities found by any VS product, it “translates” those vulnerabilities into the harmonised vulnerability categories so that the vulnerability forecast engine in component 3 of the conceptual VF model is independent of the specific VS product(s) employed in component 1 of the conceptual VF model.

The specific way in which the vulnerabilities found by a specific VS product are mapped onto the harmonised vulnerability categories is a manual procedure. This means that, before the vulnerability mapper can do anything, a person needs to take the vulnerability database of the specific VS product involved and manually map all vulnerabilities of that specific VS product onto the harmonised vulnerability categories. Fortunately, this is a once-off procedure, because every time a scan is conducted by the same VS product after the manual mapping procedure has been done, the vulnerability mapper “knows” how to map the specific VS product’s vulnerabilities found onto the harmonised vulnerability categories.

VS product vulnerability ID and description		Mapping	Harmonised vulnerability category number, icon, and name	
41	Can trace the route to the host		1	Password cracking and sniffing
			2	Network and system information gathering
62	Unpassworded Laser Jet printer found		3	User enumeration and information gathering
			4	Backdoors, Trojans and remote controlling
65	ICMP backdoor found		5	Unauthorised access to remote connections & services
			6	Privilege and user escalation
90	Sendmail syslog buffer overflow		7	Spoofing or masquerading
			8	Misconfigurations
106	Mail forgery		9	Denial-of-services (DoS) and buffer overflows
			10	Viruses and worms
110	Sendmail relaying allowed		11	Hardware specific
			12	Software specific and updates
	⋮		13	Security policy violations

Figure 6.10: VS product vulnerabilities mapped onto the harmonised vulnerability categories

In addition, the manual mapping procedure is done either by 1-to-1 mapping or by 1-to-M mapping. **1-to-1 mapping** is done by mapping one specific vulnerability in the VS database of a specific VS product onto one harmonised vulnerability category. It is also possible, however, that a specific vulnerability of a specific VS product can be mapped to many harmonised vulnerability categories. This is referred to as **1-to-M mapping**. Figure 6.10 illustrates this idea with examples. From figure 6.10 it is clear that vulnerabilities 41, 65 and 106 constitute a 1-to-1 mapping onto the harmonised vulnerability categories. Likewise, vulnerabilities 62, 90 and 110 constitute a 1-to-M mapping onto the harmonised vulnerability categories.










Event #	Host #	Date & time	Vulnerability #	Vulnerability description	Harmonised vulnerability category # and icon	
1	192.168.1.2	2003-04-15, 10:55:23	1	Anonymous FTP enabled	4 	
2		2003-04-15, 10:55:42	3	Password file not shadowed	9 	
...		
6		2003-04-15, 11:42:47	4	FTP root directory write-enabled	8 	
7		192.168.1.7	2003-04-15, 12:58:51	2	IRC server present	2 
8			2003-04-15, 12:59:58	3	Password file not shadowed	9 
...	
14	2003-04-14, 13:09:45	4	FTP root directory write-enabled	8 		
...		
56	192.168.1.41	2003-04-14, 15:41:12	1	Anonymous FTP enabled	4 	
57		2003-04-14, 15:48:00	2	IRC server present	2 	
58		2003-04-14, 15:56:04	5	Can trace route to host	2 	
...		

Figure 6.11: Scan result report with vulnerabilities mapped onto the harmonised vulnerability categories

Figure 6.11 shows the scan result report with the vulnerability mapping onto the harmonised vulnerability categories. A vulnerability of a VS product is only mapped when it is found on a particular host. For example, vulnerability number 1 found on host number 1, as shown in figure 6.11, is mapped onto harmonised vulnerability category 4.

Harmonised history database

The *harmonised history database* is the output created by the vulnerability mapper, in other words it is a database that contains the results of multiple scans conducted by the specific VS product already mapped into the harmonised vulnerability category format. Each time a vulnerability scan is conducted, a new set of harmonised vulnerability data is also created by the vulnerability mapper, and is referred to as the harmonised history database. The harmonised history database contains the entity shown in figure 6.12.

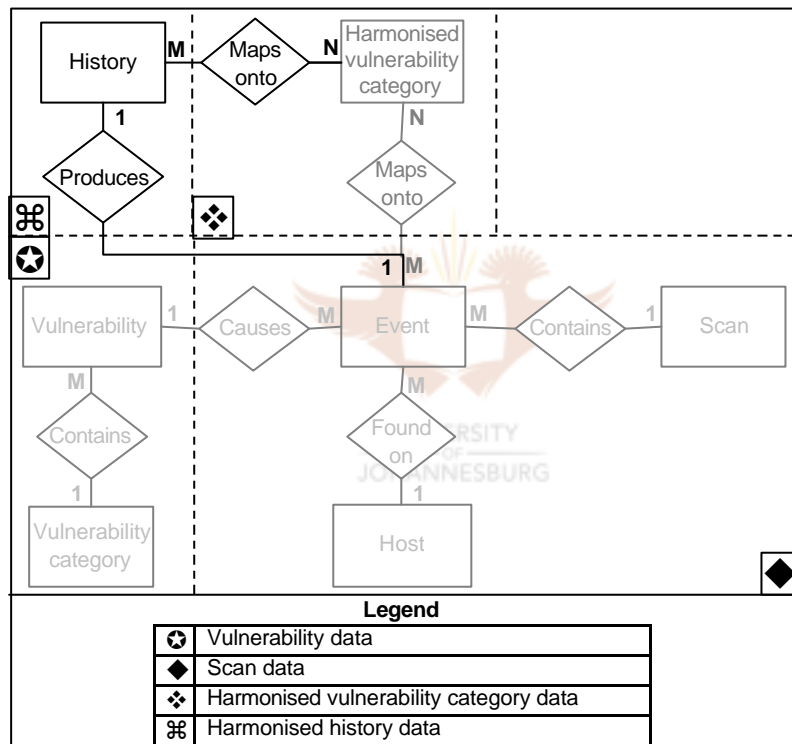


Figure 6.12: VF logical database part 4 added: harmonised history data

There is only one entity in the harmonised history database:

- **History**
 - This entity contains the history data for all the scans already conducted, mapped onto the harmonised vulnerability categories. The history entity refers to the data of all previous scans conducted. The history entity, therefore, contains the same structure as for the event entity. Each time a scan is conducted, thus, the information of the event entity is copied to the history entity, after which the event entity is cleared in order to store new scan data for the next scan to be conducted.

6.3.2.2.3 Vulnerability forecasting

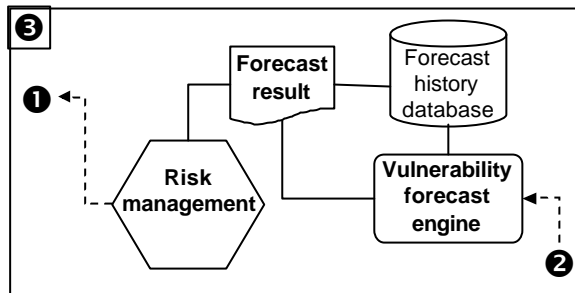


Figure 6.13: The vulnerability forecast component

Figure 6.13 represents the vulnerability forecast component of the conceptual VF model. This main component does the actual vulnerability forecasting. In summary, the output of the vulnerability mapper in component 2 of the conceptual VF model as shown in figure 6.1, namely the harmonised history database, serves as input to the vulnerability forecast engine in component 3 of the conceptual VF model. The vulnerability forecast engine attempts to predict trends or patterns, in terms of harmonised vulnerability categories, in which potential vulnerabilities could occur.

Vulnerability forecast engine

The *vulnerability forecast engine* constitutes the heart of the conceptual VF model. Intelligent techniques are used in conjunction with history scan data and history forecast data to forecast which harmonised vulnerability category or categories would potentially pose vulnerability problems in the near future. This technique will attempt to solve the addressed problems with state-of-the-art VS products as given in table 6.1. The vulnerability forecast engine will be discussed in detail in the next chapter.

Forecast history database

The forecast result is stored in the *forecast history database* along with previous forecasts carried out. The latest vulnerability forecast can be reviewed and proactive vulnerability rectification procedures can be carried out according to the vulnerability forecast. The main purpose of the forecast history database, thus, is the same as that of the harmonised history database: to serve as a repository for storing history data about vulnerability forecasts that will be used as input for the next time a vulnerability forecast is made. The forecast history database contains the entity shown in figure 6.14. The relationships in figure 6.14 denoted as M to N represent “many-to-many” relationships.

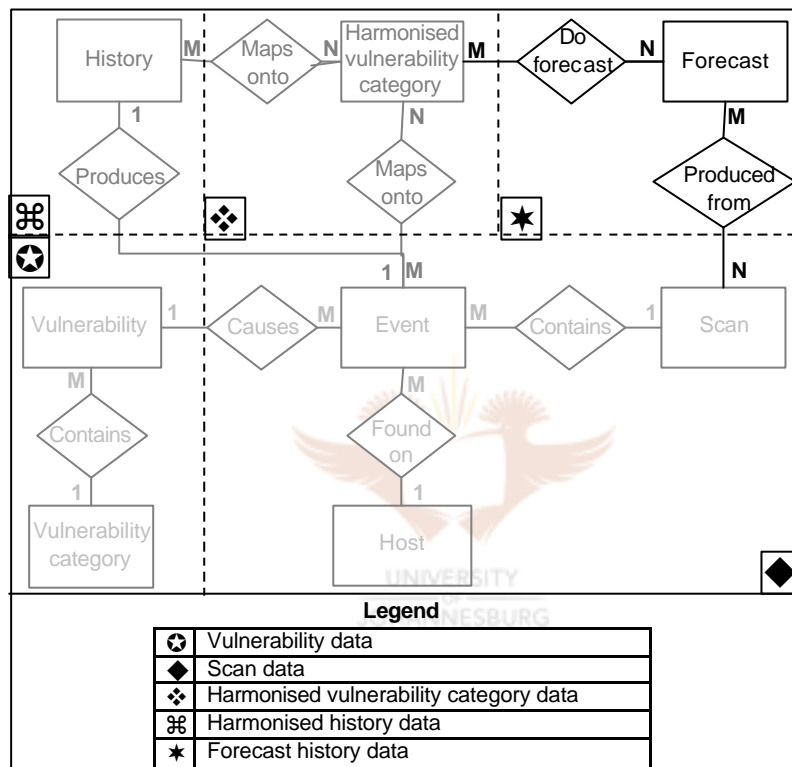


Figure 6.14: VF logical database part 5 added: forecast history data

There is only one entity in the forecast history database:

- **Forecast**

- This entity is initially empty, because no forecast has been made by then. Only from the second vulnerability forecast onwards, this entity will contain the vulnerability forecast history of all previous vulnerability forecasts that have been made. This entity contains the history data for all the forecasts already made for each harmonised vulnerability category. The forecast entity, thus, contains information of the last forecast that was made, as well as information of all previous forecasts made. The minimum fields in this entity include the *forecast ID* that serves as the primary key, the *forecast number*, and the *forecast result*.

Forecast result

A vulnerability *forecast result* is created for a specific vulnerability forecast that has been done. The goal of the report is to indicate to what extent each harmonised vulnerability category poses potential future threats to the network of hosts scanned. This enables one to take proactive action, in other words, to rectify vulnerabilities in a bid to minimise their occurrence in a future scan. The forecast result is normally archived and stored in the forecast entity in the forecast history database. The forecast result report usually contains the following information for each harmonised vulnerability category, as shown in figure 6.15:

- The vulnerability forecast number.
- The date and time when the vulnerability forecast was made.
- The harmonised vulnerability category number and description of each specific harmonised vulnerability category.
- The vulnerability forecast result for each harmonised vulnerability category.

Figure 6.15 shows vulnerability forecasts number 1 to n . Note that vulnerability forecasts 1 to $n - 1$ are considered as history forecasts and are used merely as input by the vulnerability forecast engine to effect the latest vulnerability forecast – that of forecast n . For each vulnerability forecast, i.e. vulnerability forecast number n , a vulnerability forecast result is calculated, which indicates the suggested priority of which harmonised vulnerability category should be attended to first. For example, the vulnerability forecast result for vulnerability category 8 in vulnerability forecast n suggests that *misconfiguration* vulnerabilities should receive the highest priority

where “1” indicates the highest priority and “p” indicates the lowest priority. In other words, the vulnerability forecast for the *misconfigurations* harmonised vulnerability category predicts that, if the next time a vulnerability scan is conducted and no preventative action has been taken by then, *misconfiguration* vulnerabilities might occur to such an extent that the most fatal consequences for this harmonised vulnerability category can be anticipated, compared to the consequences that might be anticipated for vulnerabilities of all the other harmonised vulnerability categories. Having this information will enable one to do risk management for the specific vulnerability forecast to rectify the vulnerabilities that are forecast for each harmonised vulnerability category.









Vulnerability forecast #	Date & time	Harmonised vulnerability category #, icon, and name	Vulnerability forecast result
1	2002-11-16, 13:25:40	9  Denial-of-services (DoS) and buffer overflows	1
		2  Network and system information gathering	2
		8  Misconfigurations	3
		11  Hardware specific	p
N	2003-04-15, 10:33:12	8  Misconfigurations	1
		7  Spoofing or masquerading	2
		1  Password cracking and sniffing	3
		10  Viruses and worms	p

Figure 6.15: A vulnerability forecast result report for vulnerability forecasts 1 to n

Risk management

The *risk management* component is not an automatic step, but rather an interactive step taken by human resources, which includes some decision-making on which and how vulnerabilities will be rectified. From figure 15 it is clear that high-risk harmonised vulnerability categories constitute those categories that were forecast with the highest expected priority. For example, harmo nised vulnerability categories 8, 7,

and 1, as shown in vulnerability forecast number n in figure 6.15, may be considered high-risk harmonised vulnerability categories. On the other hand, harmonised vulnerability category 10, as shown in vulnerability forecast number n in figure 6.15, may be considered a low-risk harmonised vulnerability category.

This would enable human resources to do risk management by providing them with more efficient information, for example on how to know where to start rectification procedures. Human resources might not necessarily decide, however, to tackle the harmonised vulnerability categories in the exact order as suggested by the vulnerability forecast. For example, vulnerability forecast number n suggests that harmonised vulnerability categories 8, 7, and 1 be attended to in that specific order. Human resources might decide, however, that, although harmonised vulnerability categories 8 and 7 are seen as high-risk categories, harmonised vulnerability category 1 might pose a higher risk to the organisation according to the organisation's needs. Harmonised vulnerability category 1, therefore, may be attended to first. To further clarify this example, suppose the organisation recently experienced devastating effects from vulnerability exploits concerned with harmonised vulnerability category 1 – *password cracking and sniffing*. Although harmonised vulnerability categories 8 and 7 are forecast to cause more vulnerability exploits to occur when the next vulnerability scan is conducted, *password cracking and sniffing* might be attended to first by human resources.

In order to get a complete picture of the conceptual VF model, all the subcomponents of the model are merged into and shown in the following section.

6.3.2.2.4 *Merging the subcomponents for the conceptual VF model*

The previous three sections discussed the three main components of the conceptual VF model in detail. These three components, when merged, comprise the conceptual VF model as shown in figure 6.16.

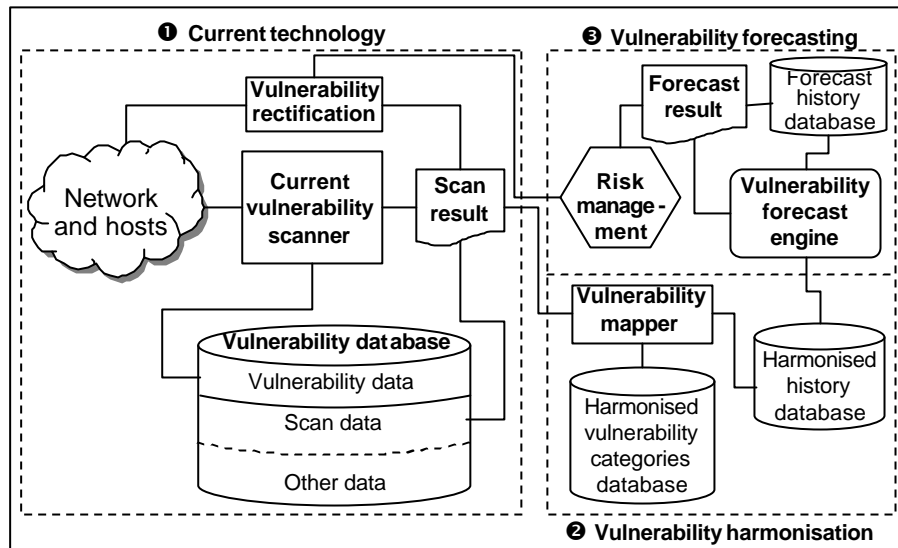


Figure 6.16: The conceptual model for vulnerability forecasting

6.4 CONCLUSION

Far from rendering existing VS products obsolete, VF is used proactively to coordinate output from existing VS products with that gleaned from intelligent techniques and history data.

The concept of vulnerability forecasting has many advantages. It saves considerable time, because instead of scans being conducted all the time to detect and rectify vulnerabilities, scans can now be conducted less frequently. Having vulnerability forecasts, vulnerability problem areas – in the form of harmonised vulnerability categories – can be attended to before they can erupt. In due course, system resources are occupied less often due to fewer scans that need to be conducted.

Using harmonised vulnerability categories along with vulnerability forecasting renders the process of doing vulnerability forecasting VS product independent. The difference in the types of vulnerability categories that various VS products scan for is therefore bridged by the use of harmonised vulnerability categories. In addition, adequate and sufficient risk management can be done due to the fact that, each time a

vulnerability forecast is done, vulnerability forecasts are made available for each harmonised vulnerability category.

In the next chapter, the researcher will elaborate on the heart of the VF model – the vulnerability forecasting engine – to demonstrate how harmonised vulnerability categories, intelligent techniques, and history data can be utilised in vulnerability forecasting.



CHAPTER 7

THE VULNERABILITY FORECAST ENGINE

7.1 INTRODUCTION

The previous chapter introduced the conceptual model for vulnerability forecasting (VF). It is necessary, however, to understand the technique that the VF model employs in more detail.

The goal of this chapter, therefore, is to discuss how the VF engine works in detail. This chapter will discuss specific steps and procedures that are used to do VF. At first, the input used to do a vulnerability forecast is discussed, followed by an explanation to show, with examples, how the VF technique works in detail.

7.2 INPUT TO THE VF ENGINE

An integral part of being able to do VF is the fact that VF makes extensive use of history data. A number of scan data sets, therefore, must be available before a vulnerability forecast can be made. These sets of scan data, however, should be in the form of harmonised history data. The scan data of a particular VS is first converted to harmonised history data, and then stored in the harmonised history database until enough harmonised history data is available to do a vulnerability forecast.

The previous chapter indicated how harmonised history data can be obtained, albeit only for one scan. Clearly, from the above, m scans need to be conducted before enough harmonised history data becomes available. Figure 7.1 illustrates harmonised history data for m scans, which serves as input to the VF engine.

Scan #	Event #	Host #	Date & time	Vulnerability #	Vulnerability description	Harmonised vulnerability category #	
Scan 1	1	192.168.1.2	2003-04-15, 10:55:23	1	Anonymous FTP enabled	4	
	2		2003-04-15, 10:55:42	3	Password file not shadowed	9	
	6		2003-04-15, 11:42:47	4	FTP root directory write-enabled	8	
	7		2003-04-15, 12:58:51	2	IRC server present	2	
	8		2003-04-15, 12:59:58	3	Password file not shadowed	9	
	14		2003-04-14, 13:09:45	4	FTP root directory write-enabled	8	
	56	192.168.1.41	2003-04-14, 15:41:12	1	Anonymous FTP enabled	4	
	57		2003-04-14, 15:48:00	2	IRC server present	2	
	58		2003-04-14, 15:56:04	5	Can trace route to host	2	
	Scan m	1	192.168.1.2	2003-04-15, 10:55:23	1	Anonymous FTP enabled	4
		2		2003-04-15, 10:55:42	3	Password file not shadowed	9
		6		2003-04-15, 11:42:47	4	FTP root directory write-enabled	8
		7		2003-04-15, 12:58:51	2	IRC server present	2
		8		2003-04-15, 12:59:58	3	Password file not shadowed	9
14		2003-04-14, 13:09:45		4	FTP root directory write-enabled	8	
56		192.168.1.41	2003-04-14, 15:41:12	1	Anonymous FTP enabled	4	
57			2003-04-14, 15:48:00	2	IRC server present	2	
58			2003-04-14, 15:56:04	5	Can trace route to host	2	

Harmonised history data – the input to the VF engine

Figure 7.1: The harmonised history data for m scans

The m sets of harmonised history data will be used to do a vulnerability forecast. To illustrate this, the next section provides a detailed discussion on how the VF technique works.

7.3 EXPLANATION OF THE VF TECHNIQUE

A typical forecast for a specific harmonised vulnerability category entitled *network and system information gathering*, may read as follows: “It is expected that a range of between x and y *network and system information gathering* vulnerabilities will be detected when the next scan is conducted.” A decision can then be made, as indicated by the **risk management** component in the VF model, as to whether or not x to y vulnerabilities for the said harmonised vulnerability category pose a significant threat to an organisation. VF can be used to compare all vulnerability categories by using intelligent techniques. Consider, for example, the forecasts for the following two vulnerability categories:

- “It is expected that a range of between 30 and 40 *privilege and user escalation* vulnerabilities will be detected when the next scan is conducted.”
- “It is expected that a range of between 5 and 9 *network and system information gathering* vulnerabilities will be detected when the next scan is conducted.”

On comparing the above vulnerability categories, it may become evident that the forecast for the *privilege and user escalation* harmonised vulnerability category warrants more urgent attention than that for the *network and system information gathering* harmonised vulnerability category, as the former poses an expected threat of “between 30 and 40” vulnerabilities – a considerably bigger threat than the vulnerability range of “between 5 and 9” forecast for the *network and system information gathering* harmonised vulnerability category.

This manner of VF can be facilitated by using a **Fuzzy Expected Interval (FEI)** [KAND 92] [SCHN 88], which forms a subset of fuzzy logic. An FEI for a specific harmonised vulnerability category represents a narrow range of typical values. This range of typical values best describes the possible number of vulnerabilities for a specific harmonised vulnerability category that can be expected to occur when a

future scan is conducted. An FEI for the above example “ x to y ” will be quoted as being range $[x, y]$.

The reasons why the researcher opted to employ an FEI include its ability to compensate for and deal with incomplete data. Incomplete data may, for instance, be generated when effecting software and hardware installations and removals. For reasons of comparison, imagine a network that contains 10 hosts. Next, assume that a new e-mail software package has been installed on 5 of the said 10 hosts and that this specific software package has been reported to contain many *privilege and user escalation* vulnerabilities. The vulnerability count for the *privilege and user escalation* harmonised vulnerability category will increase significantly after the installation. Assume, too, that the specific software package is removed at a later stage, resulting in a drop in the vulnerability count on the occasion of the very next vulnerability scan. In addition, there will always be hosts in the network that are either off-line or simply switched off, which also contributes to variations in the vulnerability count for each vulnerability scan.

In view of such fluctuations, as well as incomplete data, it would be much more prudent to forecast that the vulnerabilities expected for a specific harmonised vulnerability category belong to a specific vulnerability range, such as $[p, r]$, rather than making bold to forecast that the vulnerabilities expected for a specific harmonised vulnerability category will be equal to, say, q . For the remainder of this paper, a range such as $[p, r]$ will be referred to as a *vulnerability range*.

To explain the technique of VF, Fuzzy Expected Intervals (FEIs), which are used in the process of doing VF, will be discussed in the next section.

7.3.1 Using FEIs for VF

An FEI is calculated for each harmonised vulnerability category. Consider that m scans have been conducted to obtain harmonised history data. Also consider only one specific harmonised vulnerability category K over the m scans for now. Calculating the FEI for harmonised vulnerability category K , and subsequently for all other harmonised vulnerability categories, requires the steps depicted in figure 7.2. These steps are based on work done by Schneider [SCHN 88].

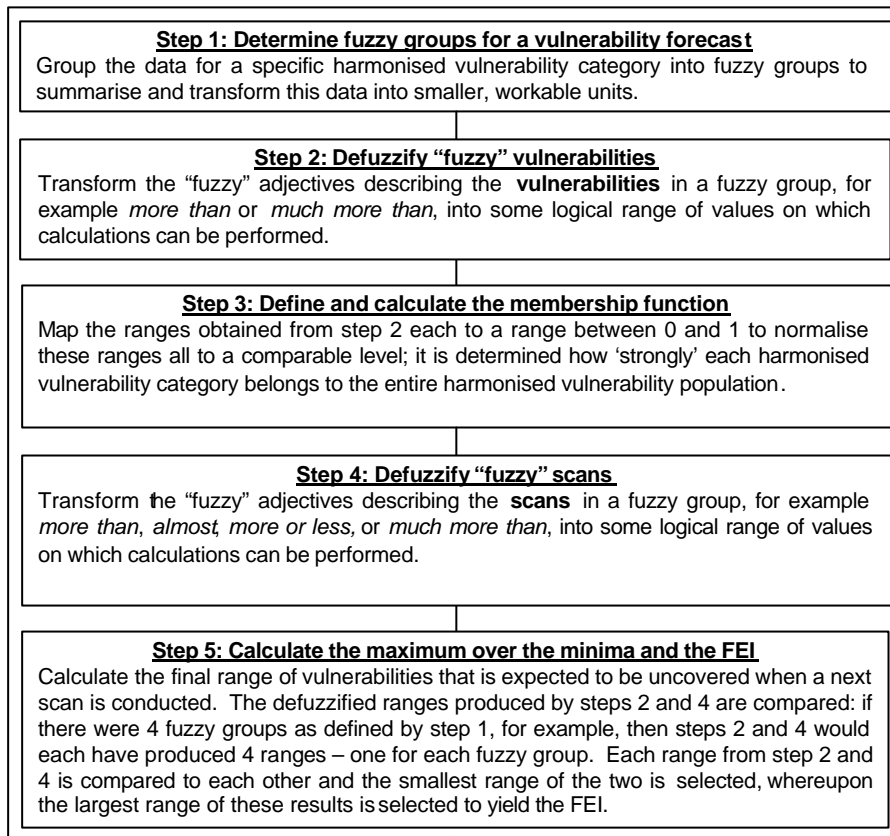


Figure 7.2: The 5 steps for determining the FEI for each harmonised vulnerability category

Each of the steps mentioned in figure 7.2 is discussed in detail in the sections that follow.

7.3.1.1 Step 1: Determine fuzzy groups for a vulnerability forecast

As a starting point, and before any calculations can be done, it is critical to know how the population is distributed across the harmonised history data obtained. In other words, the population is distributed into certain fuzzy groups. The term “population” is used to refer to the entire range of all possible values in the harmonised history data.

Determining these fuzzy groups is an intuitive exercise and can be effected, for example, by examining the history data for harmonised vulnerability category **K** over

m scans. The specific number of fuzzy groups to be determined is also arrived at intuitively – it depends on how closely the data is related. Each of the m scans must be allocated to a specific fuzzy group. The idea is to group together the harmonised history data of those scans that closely relate to one another, for harmonised vulnerability category K . In this way, the data in the m scans is summarised and lumped into fuzzy groups [KAND 82]. Table 7.1 shows examples of fuzzy groups.

Table 7.1: The fuzzy groups formed for harmonised vulnerability category K

Fuzzy group	Fuzzy group description
A	In 3 scans, 20 to 25 vulnerabilities were found.
B	In 3 to 5 scans, more or less 20 vulnerabilities were found.
C	In 4 to 5 scans, almost 80 vulnerabilities were found.

Note that the scans grouped together in fuzzy group A fall *exactly* within the [20, 25] range. There is, in other words, nothing “fuzzy” about this range. Fuzzy groups B and C, on the other hand, contain the adjectives **more or less** and **almost** respectively, which render these ranges fuzzy, and not crisp.

7.3.1.2 Step 2: Defuzzify “fuzzy” vulnerabilities

In the process of calculating an FEI, working with fuzzy values is precluded, as the calculations in question can only be effected through crisp logic. The adjectives **more or less** and **almost** need to be converted into ranges – they need to be defuzzified. What exactly, for instance, does “more or less 20” mean in fuzzy group B? In fuzzy group A, an exact range is given, namely [20, 25]. The adjective “more or less 20” can be converted to such an exact range by converting it to a specific value. This process of converting adjectives into a range with a lower and an upper bound can be effected by means of a mapping table [SCHN 88], as shown in table 7.2.

Table 7.2: Mapping table

Adjective	Lower bound	Upper bound
Almost	$x - 15\%$	$x - 3$
More or less	$x - 2$	$x + 10\%$

How exactly are the formulae for the lower and upper bounds determined for each of the adjectives listed in table 7.2? As with creating the fuzzy groups, the mapping table is also constructed intuitively by studying the harmonised history data gleaned

from the m scans as follows: As far as the adjective **almost** is concerned, the term “almost” implies that neither the lower nor the upper bound reaches the specific value that is being quoted. Consider again, for example, fuzzy group C: “*In 4 to 5 scans, almost 80 vulnerabilities were found.*” When looking at specific scans out of the m scans, 4 to 5 scans might happen to fall in the vulnerability range [40, 50], as an example. The lower bound of these scans is 40, and $40 \sim 50 - 10\%$. Consider this calculation, $50 - 10\%$, and substitute “50” for “ x ”. Having done so, the “lower-bound formula” for “almost” in the mapping table yields $x - 15\%$. The maximum (upper bound) is calculated in the same way. Similarly, the remainder of the mapping table is constructed for the adjective **more or less**.

The mapping table is applied in the following way. The data in fuzzy group A from table 7.1 is already in a range format, with a lower and an upper bound, namely [20, 25]. “More or less” 20 in fuzzy group B, however, still needs to be converted, using the mapping table provided in table 7.2. If the adjective “more or less” could be ignored for the moment, “20” could, in range format, be written as [20, 20], indicating that both the lower and upper bounds of the range are 20, so that $20 = [20, 20]$. The formula (according to the mapping table) to “remove” the “more or less” section in fuzzy group B is $x - 2$ for the lower bound and $x + 10\%$ for the upper bound of the range. This, in turn, means that x simply needs to be substituted for the range [20, 20] (lower and upper bounds respectively) in order to calculate the “converted” lower and upper bounds. In this way, the “converted” range for fuzzy group B becomes $[20 - 2, 20 + 10\%] = [18, 22]$. After having effected the latter defuzzification process, fuzzy group B actually reads as follows: “*In 3 to 4 scans, 18 to 22 vulnerabilities were found.*” In the same way, all the adjectives of the other fuzzy groups can be converted into ranges.

To sum up, table 7.3 below indicates the distribution of the scans as they were allocated to each fuzzy group, as well as the defuzzified ranges for the current example.

Table 7.3 : Distribution of scans and defuzzified ranges

Fuzzy group	Distribution of scans	Vulnerabilities found (defuzzified)
A	3 scans become [3, 3]	[20, 25]
B	3 to 5 scans become [3, 5]	[18, 22]
C	4 to 5 scans become [4, 5]	[68, 77]

These defuzzified ranges now need to be converted again, using a membership function. The latter conversion is necessary in order to obtain a normalised view of all the vulnerability ranges. This process will be discussed in the next section.

7.3.1.3 Step 3: Define and calculate the membership function

The membership function simply expresses each of these defuzzified vulnerability ranges as a range between 0 and 1. In other words, the membership function expresses the grade of membership [KABY 78] for each defuzzified vulnerability range compared with the entire population. Traditionally, the grade of membership 1 is assigned if the vulnerability range completely and fully belongs to the entire population, whilst 0 is assigned to a vulnerability range that does not belong to the entire population at all. The greater the degree to which a vulnerability range belongs to the entire population, the closer it is to a grade of membership to the entire population.

The membership function for this example would be constructed as shown in figure 7.3. As in the case of the fuzzy groups and the mapping table, the membership function is created intuitively. Assume that it is evident from the harmonised history data that not one of the m scans ever uncovered more than 90 vulnerabilities. This membership function, therefore, indicates that 90 is the absolute maximum that will ever be reached for harmonised vulnerability category K .

$$c(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ x / 90 & \text{if } (0 < x < 90) \\ 1 & \text{if } x \geq 90 \end{cases}$$

Figure 7.3: Membership function

The result of this membership function for each fuzzy group will be referred to as c 's, as shown in table 7.4. For example, the c range for the range [20, 25] is calculated

first for the lower bound and then for the upper bound. This culminates in the c range [0.222, 0.278] for fuzzy group A, as shown in table 7.4. The latter process is repeated for the rest of the fuzzy groups.

Table 7.4: Transforming the defuzzified values using the membership function $c(x)$

Fuzzy group	Distribution of scans	c's
A	[3, 3]	[0.222, 0.278]
B	[3, 5]	[0.2, 0.244]
C	[4, 5]	[0.756, 0.889]

The distribution of scans must now also be defuzzified to a fuzzy measure between 0 and 1. In other words, finding the fuzzy measure for each c range implies determining the degree to which a specific range of vulnerabilities belongs to the entire population. This process will be discussed in the next section.

7.3.1.4 Step 4: Defuzzify “fuzzy” scans

The method of calculating fuzzy measures over a certain distribution of the population is described by Schneider [SCHN 88]. As in the previous step, a fuzzy measure for the scan distribution of each fuzzy group will be calculated. These results will be referred to as m's. Schneider [SCHN 88] uses equations 1 and 2, below, to calculate the m's for each fuzzy group, where n is the number of fuzzy groups identified and j is the current fuzzy group in the calculation of **LB_j** (the m for the **lower bound** of fuzzy group j) and **UB_j** (the m for the **upper bound** of fuzzy group j).

$$\text{Equation 1: } LB_j = \frac{\sum_{i=j}^n \text{MIN } [p_{i1}, p_{i2}]}{\sum_{i=j}^n \text{MIN } [p_{i1}, p_{i2}] + \sum_{i=1}^{j-1} \text{MAX } [p_{i1}, p_{i2}]}$$

$$\text{Equation 2: } UB_j = \frac{\sum_{i=j}^n \text{MAX } [p_{i1}, p_{i2}]}{\sum_{i=j}^n \text{MAX } [p_{i1}, p_{i2}] + \sum_{i=1}^{j-1} \text{MIN } [p_{i1}, p_{i2}]}$$

Having used these two equations to calculate each of the m 's for each fuzzy group, the results are as shown in table 7.5 below:

Table 7.5: Results for the m 's and c 's

Fuzzy group	m 's	c 's
A	[1.0, 1.0]	[0.222, 0.278]
B	[0.7, 0.769]	[0.2, 0.244]
C	[0.333, 0.455]	[0.756, 0.889]

The m 's and c 's in table 7.5 above can be explained as follows, for example, for groups A and C:

- For fuzzy group A, the defuzzified vulnerability range [0.222, 0.278] belongs to the entire scan population, since [1.0, 1.0] is exactly 1, and since 1 belongs to the entire scan population. The defuzzified vulnerability range [0.222, 0.278], thus, is included in all of the scans owing to its membership of exactly 1 to the entire population.
- For fuzzy group C, the defuzzified vulnerability range [0.756, 0.889] belongs [0.333, 0.455] to the scan population, and [0.333, 0.455], in a range between 0 and 1, means that this vulnerability range includes only [0.333, 0.455] times the entire scan population. The defuzzified vulnerability range [0.756, 0.889], thus, is included in very few of the m scans, owing to its membership of [0.333, 0.455] to the entire population of m scans.

The ultimate aim is to find a single range that would serve as the FEI for this specific harmonised vulnerability category. To find one specific range from table 7.5, the median of all those ranges must be calculated [SCHN 88]. This is done by calculating the maximum over the minima of all the ranges shown in table 7.5. This also constitutes the final step in the process, to be discussed in the next section.

7.3.1.5 Step 5: Calculate the maximum over the minima and the FEI














For the final step in this process, the $\text{MAX}(\text{MIN}(m_i, c_i))$ is calculated, using theorems of Schneider [SCHN 88]. The result for this example would yield [0.333, 0.455]. The latter value should be multiplied by **90** to express the range [0.333, 0.455] in “number of vulnerabilities”. Bear in mind that the number **90** was used in this

example to express the population as the absolute maximum number of vulnerabilities anticipated and reflected in the membership function. This will yield an FEI of [30, 41], which, in turn, serves to forecast that the next time the specific VS used in this example is used to conduct a scan, it will uncover 30 to 41 vulnerabilities for harmonised vulnerability category *K*.

7.3.2 FEI for each harmonised vulnerability category

The above vulnerability forecast was effected for harmonised vulnerability category *K* only. It can, however, also be effected for the remainder of the harmonised vulnerability categories, which will each yield an FEI. Table 7.6 depicts such an example of the vulnerability forecasts for harmonised vulnerability categories 1 to 13.

Table 7.6: An example FEI calculated for each harmonised vulnerability sorted in order of highest to lowest priority

Harmonised vulnerability category number, icon, and name		Vulnerability forecast result
9	 Denial-of-services (DoS) and buffer overflows	[20, 31]
3	 User enumeration and information gathering	[23, 27]
8	 Misconfigurations	[8, 24]
4	 Backdoors, Trojans and remote controlling	[19, 22]
5	 Unauthorised access to remote connections & services	[10, 21]
1	 Password cracking and sniffing	[14, 19]
6	 Privilege and user escalation	[0, 7]
2	 Network and system information gathering	[5, 5]
12	 Software specific and updates	[4, 5]
7	 Spoofing or masquerading	[2, 6]
11	 Hardware specific	[4, 4]
13	 Security policy violations	[3, 5]
10	 Viruses and worms	[0, 4]

7.4 CONCLUSION

The question could be asked: What do we stand to gain from obtaining such forecast results? The answer to this question is that forecast results could give us an indication of the number of potential vulnerabilities an organisation could expect to have uncovered for each of its vulnerability categories during future scans. In this way, an organisation could perform risk management to better prioritise its vulnerability problem categories.

The problem areas of particular interest to the researcher of this research project included the duration of vulnerability scans, as well as the possible degradation of system performance during such scans. How can VF be used to minimise the impact of these problems? VF helps to reduce the number of vulnerability scans required over time by increasing the intervals between successive scans. Instead of having to conduct scans on a daily basis, for instance, scans may now be required on a weekly basis only. This reduction in the number of scans can, moreover, be accomplished without forfeiting anything as far as the exposure or rectifying of vulnerabilities is concerned. In this way, not only the overall number of scans required for efficiency is reduced, but, consequently, also the threat of system degradation during scans.

The next chapter will demonstrate the working of VF by means of actual scan data and a prototype implementation.



CHAPTER 8

A PROTOTYPE FOR VULNERABILITY FORECASTING

8.1 INTRODUCTION

The thesis culminates in this chapter through a demonstration of the deployment of the VF model in a prototype implementation called the **VF Prototype**. The VF Prototype is not a complete implementation of the VF model. However, it implements the biggest part of the VF model that is essential to demonstrate that the VF model works. The sections that follow will state the aim of the VF Prototype as well as what part of the VF model is implemented by the prototype.

8.1.1 The aim of the prototype

The VF Prototype has as its primary aim to demonstrate that, by having gathered sufficient vulnerability history data, a forecast can be made about how and to what extent vulnerabilities will occur in the future. The VF Prototype can therefore be used as part of a risk management programme that would provide human resources with sufficient information, enabling them to deal more successfully with vulnerabilities which will occur when a vulnerability scan is conducted in the future. The prototype is specifically concerned with producing a vulnerability forecast range for each specific harmonised vulnerability category.

Although the vulnerabilities of a specific VS product should be classified into the harmonised vulnerability categories – ultimately by the VF Prototype itself – the current version of the prototype relies on the input of human resources as a once-off classification exercise for each VS product that is used by the VF model. In addition, the VF Prototype will use only one specific VS product for doing vulnerability forecasting.

8.1.2 The VF model and the prototype

The scope of implementation of the VF Prototype in terms of the VF model is graphically depicted in figure 8.1 by the dark black line as being mostly concerned with the vulnerability forecasting process itself, and not with the gathering of history data. The vulnerability rectification, risk management, and current VS components have been specifically left out of the scope of implementation of the VF Prototype.

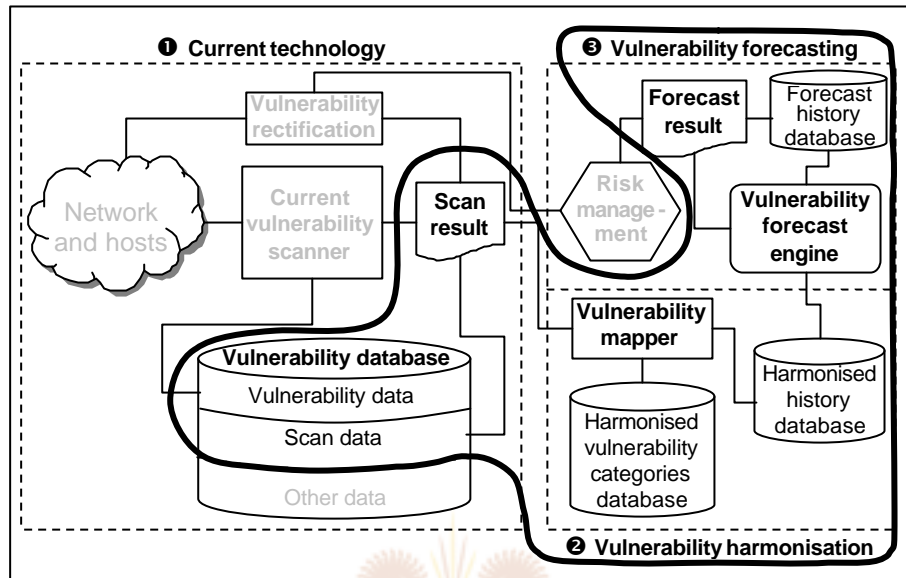


Figure 8.1: The VF Prototype's scope in terms of the VF model as indicated by the dark black line

In addition, the dark black line in figure 8.2 shows the scope of implementation of the VF Prototype in terms of the database structure of the VF model as defined in chapter 6. The database implemented in the VF Prototype, thus, differs somewhat from that specified for the VF model. For one, it does not contain a history entity as shown in figure 8.2. However, vulnerability scans that are conducted accumulate in the event entity rather than in a separate history entity as shown for the VF model. The VF Prototype also refers differently to some of the entities and attributes as discussed in chapter 6. The reason for this is that, for the purpose of demonstrating the VF Prototype, a specific VS product – CyberCop Scanner – has been used.

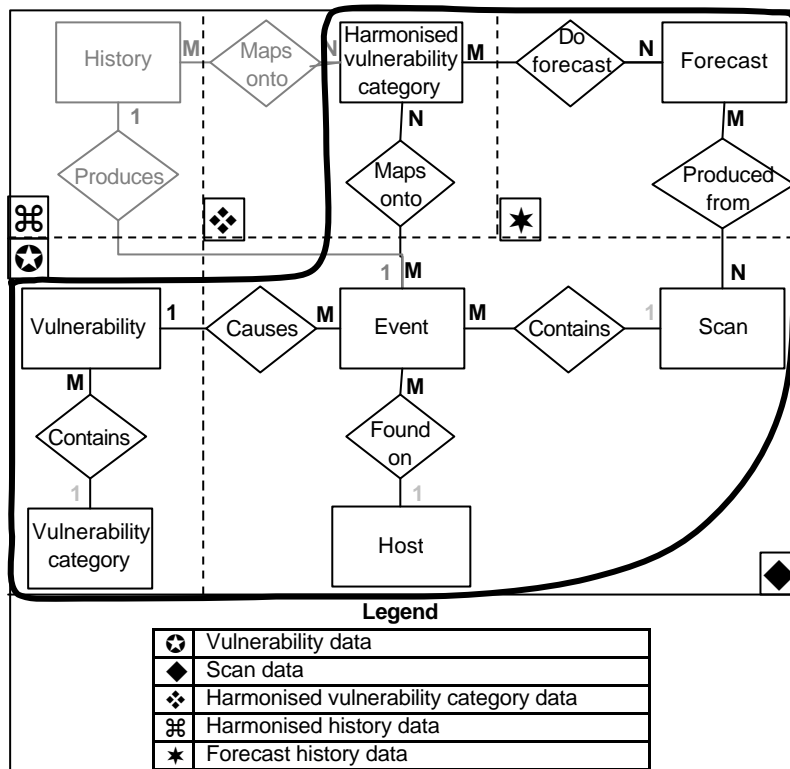


Figure 8.2: The VF Prototype's scope in terms of the VF model as indicated by the dark black line

Figure 8.3 shows a screenshot of the relational database schema as defined in the VF Prototype.

The next section explains the development of the VF Prototype. A discussion of the choice of tools will start the explanation of the prototype. Thereafter the functional requirements that were set out will be explained and an architectural overview of the prototype will be given. A word on installing the VF Prototype will then be given, but the detailed installation procedure is given in Appendix A. This is followed by a section on the operation of the VF Prototype with discussions on the specific scan scenario in which the prototype was tested, setting up parameters of the prototype before it can execute, and a detailed discussion on using the prototype software. Finally, the chapter will conclude with remarks and findings regarding the VF Prototype.

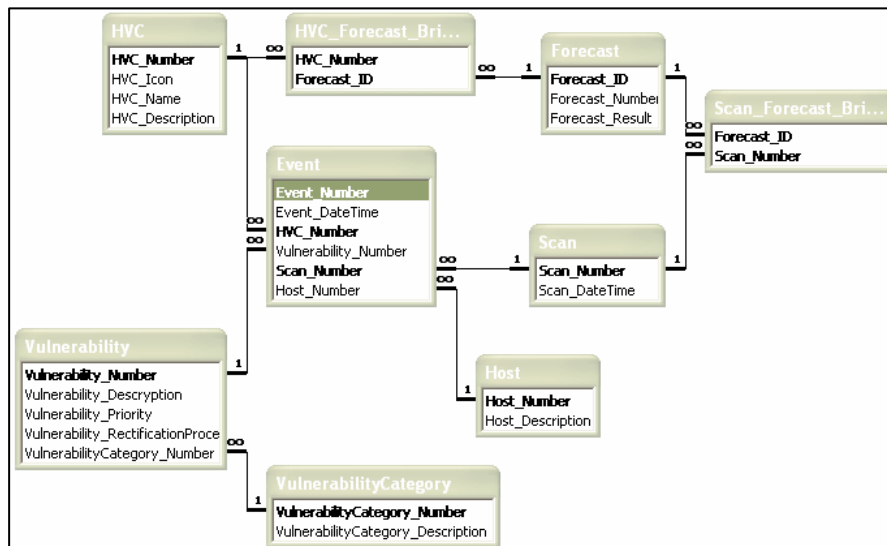


Figure 8.3: Relational schema of the database implementation in the VF Prototype

8.2 DEVELOPMENT OF THE VF PROTOTYPE

In order to achieve the aim of the VF Prototype, the researcher followed a combined approach by using existing VS products as tools and integrating them in the development of the VF Prototype.

The development tools used for designing the VF Prototype consequently need to be considered. The development of the VF Prototype was strongly supported by a number of tools. The development took place in two phases: a design phase and an implementation phase. During the design phase, Microsoft Access was used to study the database structure of current VS products, as well as to design the database structure of the VF Prototype. Microsoft Excel in conjunction with Visual Basic for Applications were also used in this design phase to draw graphs and develop preliminary results.

During the implementation phase, the VF Prototype was created. This phase was developed using Microsoft Visual Basic. The source code for the VF Prototype is given in Appendix B.

8.3 INSTALLATION OF THE VF PROTOTYPE

For installing the VF Prototype software and additional software components needed for the prototype to run, refer to Appendix A. Note that Appendix A also contains important information about conventions used in this chapter.

After the VF Prototype software has been installed successfully, it is ready to run as described in the next section.

8.4 OPERATION OF THE VF PROTOTYPE

Before the operation of the VF Prototype software is discussed in detail, a background to the VF Prototype is given, followed by a description of a scan scenario, which was staged specifically for testing the VF Prototype software.

8.4.1 Background to the VF Prototype

The researcher used the output of current VS products, i.e. that of CyberCop Scanner, as input to the VF Prototype. On completion of a vulnerability scan, the typical output of a VS product such as CyberCop Scanner is produced in the form of a report. Such reports often also contain extensive information on how to rectify the vulnerabilities that were uncovered. Typical information to be contained in a scan report would be the host IP address, a unique vulnerability ID, a vulnerability description, security concerns, rectification procedures and the vulnerability category to which the specific vulnerability belongs. An example of one such vulnerability report produced after a scan is depicted in table 8.1. After a number of scans have been conducted, a history of scan data is generated. A detailed CyberCop Scanner report of one such scan is shown in Appendix C.

Some VS products, i.e. CyberCop Scanner, additionally categorise the vulnerabilities that they can scan for into vulnerability categories. Table 8.2 shows the 31 vulnerability categories that CyberCop Scanner specifically defines. CyberCop Scanner uses a vulnerability database containing signatures for approximately 700 vulnerabilities. See Appendix D for a detailed layout of the vulnerabilities found in the vulnerability database of CyberCop Scanner.

Table 8.1: Example of a vulnerability uncovered during a scan

Host IP address	192.168.1.12
Vulnerability ID	30006
Description	Remote Access Service (RAS) detected on the host. The RAS lets remote users use a telephone line and a modem to dial into an RAS server and tap the resources of its network.
Security concerns	A person could be using an RAS to gain access to a network from a remote location. This essentially creates a "backdoor" into a network, which can, for example, bypass the firewall of that network.
Rectification procedures	Investigate this host to identify if it is indeed an approved RAS host. If so, there may be ways in which further to secure the host, for example, the RAS can be configured to establish a connection by merely calling back a user automatically, thereby ensuring that the user's telephone number which is used to gain access via this RAS host is known to the RAS server.
Vulnerability category	Remote connections and services.

Table 8.2: CyberCop vulnerability categories

Vulnerability category No.	CyberCop vulnerability category name
1	Information gathering and reconciliation
2	File-transfer protocols
3	Hardware peripherals
4	Backdoors and misconfigurations
5	SMTP and mail transfer
6	Remote procedure call services
7	Networked file systems
8	Denial-of-service attacks
9	Password guessing/grinding
10	World Wide Web, HTTP and CGI
11	Network protocol spoofing
12	Packet filter verification tests
13	Firewalls, filters and proxies
14	Authentication mechanisms
15	General remote services
16	SMB/NetBIOS resource sharing
17	Domain name system and BIND
18	Windows NT – network vulnerabilities
19	Not used
20	SNMP/network management
21	Network port scanning
22	Windows NT – browser zone policy
23	Windows NT – privilege enumeration
24	Windows NT – local system policy
25	Windows NT – auditing and password policy
26	Windows NT – information gathering
27	Intrusion-detection system verification
28	Windows NT – service packs (SPs) and hot fixes (HFs)
29	Windows NT – third-party software
30	Windows NT – services
31	Windows NT – remote access server

In the sections that follow, it will become evident how the VF Prototype maps CyberCop Scanner's 31 vulnerability categories onto the harmonised vulnerability categories. The following section, however, shows a specific scan scenario where CyberCop Scanner was used to gather history scan data that would serve as input to the VF Prototype.

8.4.2 The scan scenario

First, the specific platform for staging the scan scenario is discussed. Then a discussion on the scenario itself is given.

8.4.2.1 The platform

The VF Prototype software was tested on the following platform specifications:

- Hardware
 - Intel Pentium III 750 MHz.
 - 128MB main memory (primary storage).
 - 10GB hard disk drive (secondary storage).
 - 3COM network interface card (NIC) connected to a local area network (LAN).
- Software
 - Microsoft Windows 2000 Professional Service Pack 2.
 - CyberCop Scanner 5.5 [CYBE 03].
 - The VF Prototype [VENT 03, VISS 03].

8.4.2.2 The scenario

CyberCop Scanner was connected to a staged network scan scenario as an experiment. The network used for the experiment consisted of 59 workstations in a multi-platform environment, as depicted in figure 8.4.

In order to collect sufficient history data for the experiment, 15 scans were conducted over regular time intervals, i.e. on a daily basis. Each scan conducted for this network scenario lasted several hours.

The steps taken for this experiment were as follows: CyberCop Scanner was configured to scan the 59 workstations. The scan process was then initiated on the

CyberCop Scanner computer. CyberCop Scanner then scanned the first host, by testing all 700 vulnerability signatures in its vulnerability database in a bid to report which vulnerability signatures were found to match on this host. The latter process was repeated on the remaining 58 hosts. In total, approximately 41 300 vulnerabilities were checked for – 700 vulnerabilities for each of the 59 hosts. Lastly, a report was generated, listing all matched vulnerabilities – see Appendix C.

Figure 8.5 indicates the number of matched vulnerabilities uncovered for each of the CyberCop Scanner vulnerability categories during scan 1 over the 59 hosts. Appendix E shows the graphs of all 15 scans.

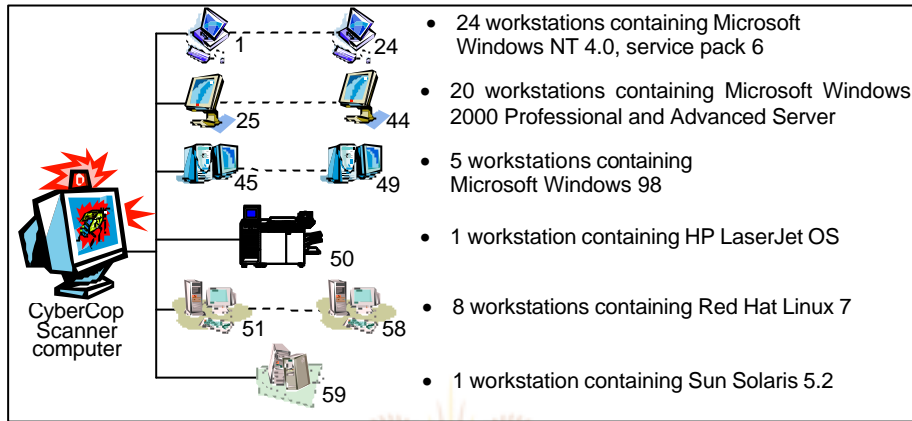


Figure 8.4: CyberCop Scanner network scan scenario

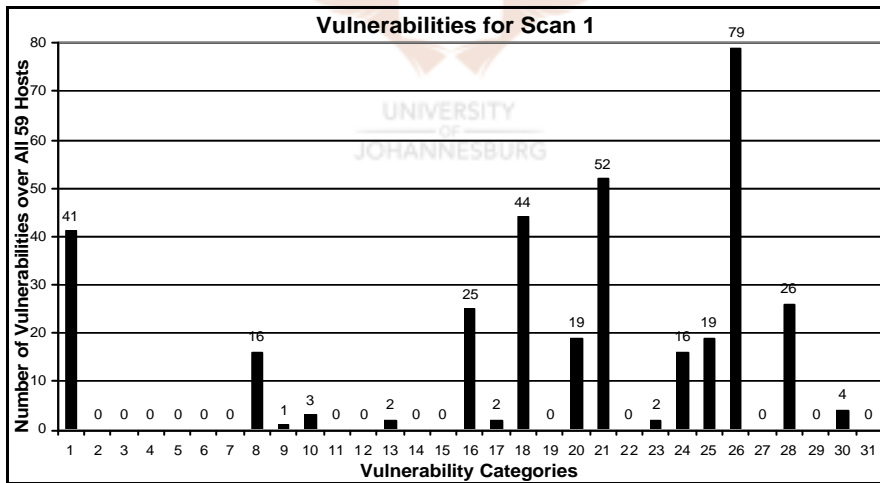


Figure 8.5: Vulnerabilities uncovered by CyberCop Scanner during scan 1 for each CyberCop Scanner vulnerability category

In the sections that follow, the VF Prototype is executed. Before the VF Prototype can deliver any results, however, it has to be set up– this is shown in the next section.

8.4.3 Setting up the VF Prototype parameters

In order to run the VF Prototype software, click *Start* → *Run* in Windows. Type the command `C:\VF\VFPrototype.exe` into the **Open** field of the **Run** window as shown in figure 8.6.

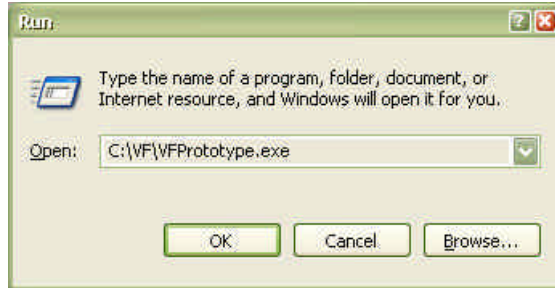


Figure 8.6: Running the VF Prototype software

The **Vulnerability Forecasting (VF) Prototype** main window should appear as shown in figure 8.7 without any data entered initially. From this window, all of the setup parameters and forecasting functions are accessible. In the **Specify the directory that contains all the scanning databases** box, enter `C:\VF\Scans` to specify where the history data resides that was produced by CyberCop Scanner. In this scan scenario, the history data comprises the 15 scans conducted by CyberCop Scanner over a period.

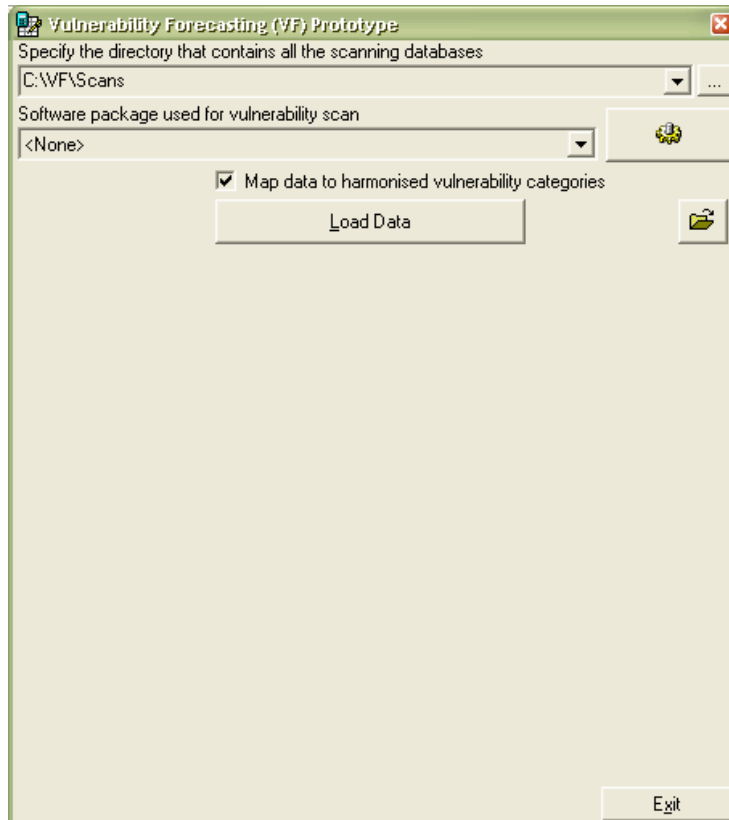




Figure 8.7: The Vulnerability Forecasting (VF) Prototype main window

Now click on the **options** button  to the right of the **Software package used for vulnerability scan** box in order to specify the VF Prototype options. Note that the term “software package” denotes the specific “VS product” used by the VF Prototype. The VS Prototype is able to use data from more than one different VS product. For the purpose of this scenario, only one VS product has been used – CyberCop Scanner. After clicking on the **options** button , the **VF Prototype – Options** window should appear as shown in figure 8.8 – initially empty.

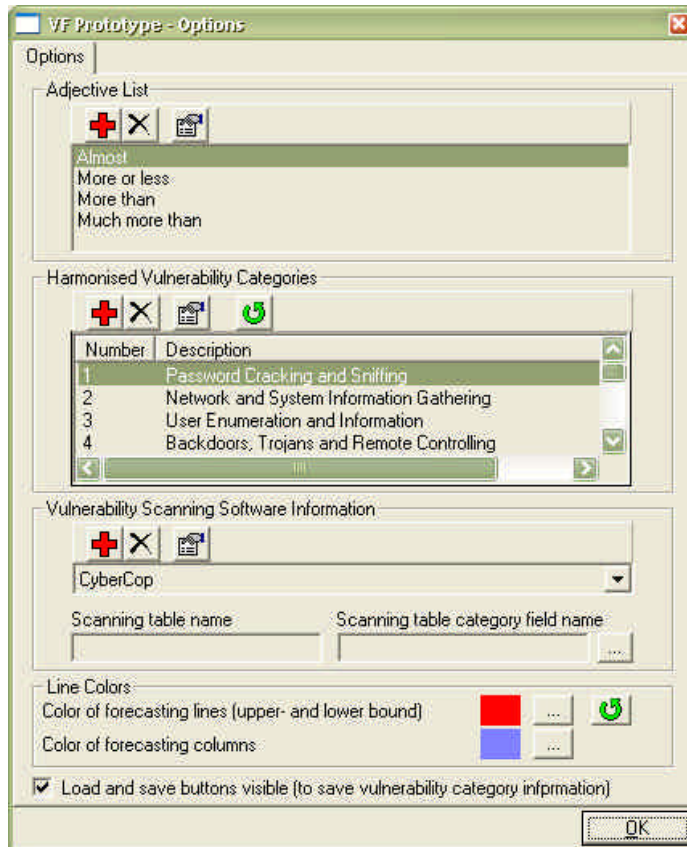



Figure 8.8: The VF Prototype – Options window

8.4.3.1 Setting up the Adjective List

The **Adjective List** is a list that constitutes a measure of fuzziness, for example “almost”, “more or less”, or “more than”, as described in chapter 7. To add the adjective “almost” to the adjective list, click on the **Add Adjective** button  and enter the new adjective in the **Adding Adjective** input box as shown in figure 8.9.

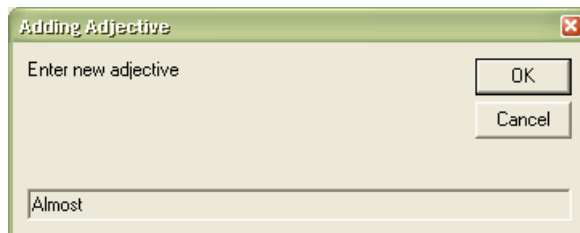




Figure 8.9: The Adding Adjective input box

After entering the description of an adjective, i.e. `Almost`, click on **OK**. Repeat this process for all adjectives that must be added. If an adjective needs to be deleted, click on the adjective in the **Adjective List** and click on the **Delete Adjective** button **X**. If an existing adjective needs to be edited, click on the adjective in the **Adjective List** and click on the **Edit Adjective** button .

8.4.3.2 Setting up the Harmonised Vulnerability Categories

In the **Harmonised Vulnerability Categories** list, as shown in figure 8.8, the harmonised vulnerability categories as discussed in chapter 4 are entered. This is done in the same way as explained for the **Adjective List**. The **Default Categories** button  simply serves as a shortcut to enter hard-coded harmonised vulnerability categories. It can be ignored for the purpose of this explanation.

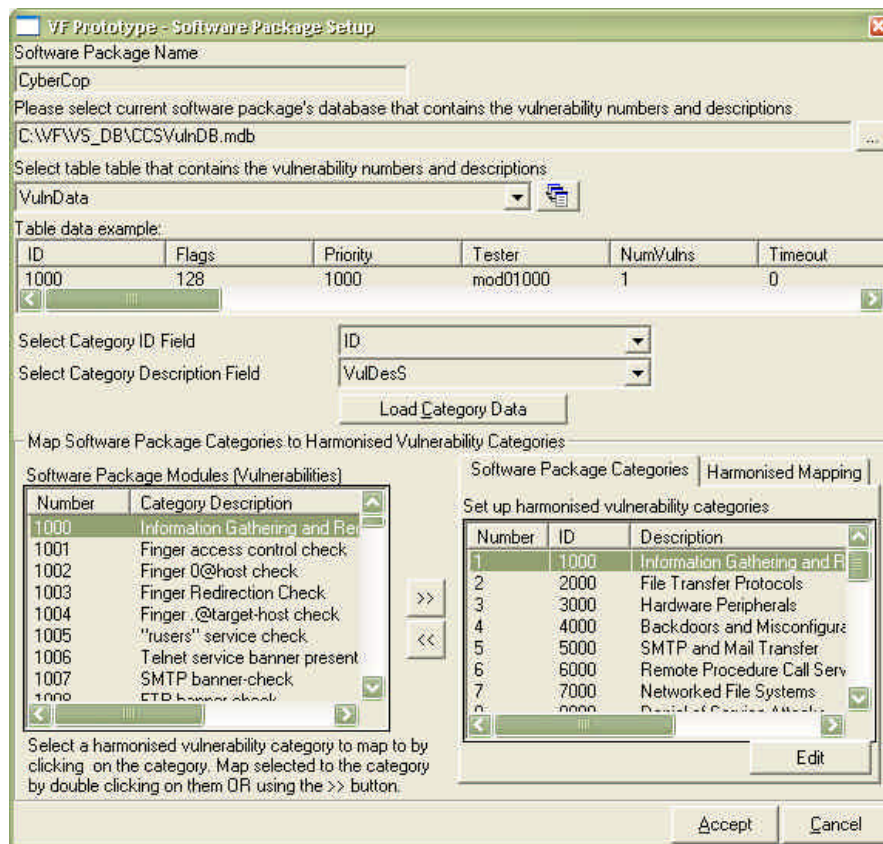



Figure 8.10: The VF Prototype – Software Package Setup window

8.4.3.3 Setting up the VS product used

The **Vulnerability Scanning Software Information** frame, as shown in figure 8.8, allows the user to set up a specific VS product that is used to conduct VS scans – CyberCop Scanner in this scenario. Click on the **Add New** button  to set up CyberCop Scanner for use by the VF Prototype. Clicking on this button should open the **VF Prototype – Software Package Setup** window, as shown in figure 8.10.


8.4.3.3.1 Specifying the VS product name

In the **Software Package Name** box, as shown in figure 8.10, enter a name, e.g. CyberCop. The software package or VS product used will be referred to as the CyberCop Scanner from here onwards.

8.4.3.3.2 Specifying the vulnerability database path

In the **Software package database** box, enter the path containing the vulnerability database of the CyberCop Scanner, e.g. `C:\VF\VS_DB\CCSVulnDB.mdb`.

8.4.3.3.3 Specifying the tables used

Click on the **load tables** button  in order to load all the tables from the **CCSVulnDB.mdb** database to be populated into the drop-down list next to this button. Note that, as the form is being filled in, more buttons and boxes become available. The specific table containing the vulnerability data of the CyberCop Scanner is the **VulnData** table. A **Table data example** is loaded to show the user what the data in the **VulnData** table looks like for each field in the table. Not all the fields in such a table will be used by the VF Prototype – only a vulnerability ID field and a vulnerability description field is necessary. A user, therefore, will now be able to easily make a choice between the latter two fields in this table to be selected in the **Select Category ID Field** drop-down list and the **Select Category Description Field** drop-down list, respectively. In this scenario, **ID** and **VulDesS** are the two fields to be selected as the vulnerability ID and the vulnerability description fields, respectively.

Once the two fields have been selected, click on the **Load Category Data** button in figure 8.10 to display lists that would enable the user to specify firstly the categories

as defined by the CyberCop Scanner, and secondly how the vulnerabilities of the CyberCop Scanner are mapped onto the harmonised vulnerability categories.

8.4.3.4 Specifying the software package categories

The CyberCop Scanner's categories are stored in the same table as the vulnerabilities. A category or a vulnerability, thus, is seen as the same object – a so-called software package module – in the CyberCop Scanner vulnerability database. The way in which a category is differentiated from a vulnerability, however, is by the ID **Number**. Unfortunately, CyberCop Scanner does not store its vulnerability data and scan data in a relational database. Instead, vulnerabilities and vulnerability categories are stored in a single table. CyberCop Scanner, therefore, uses the following scheme to distinguish between a vulnerability and a vulnerability category: an ID **Number**, where the **Number** is a multiple of 1 000, is a vulnerability category and the rest are

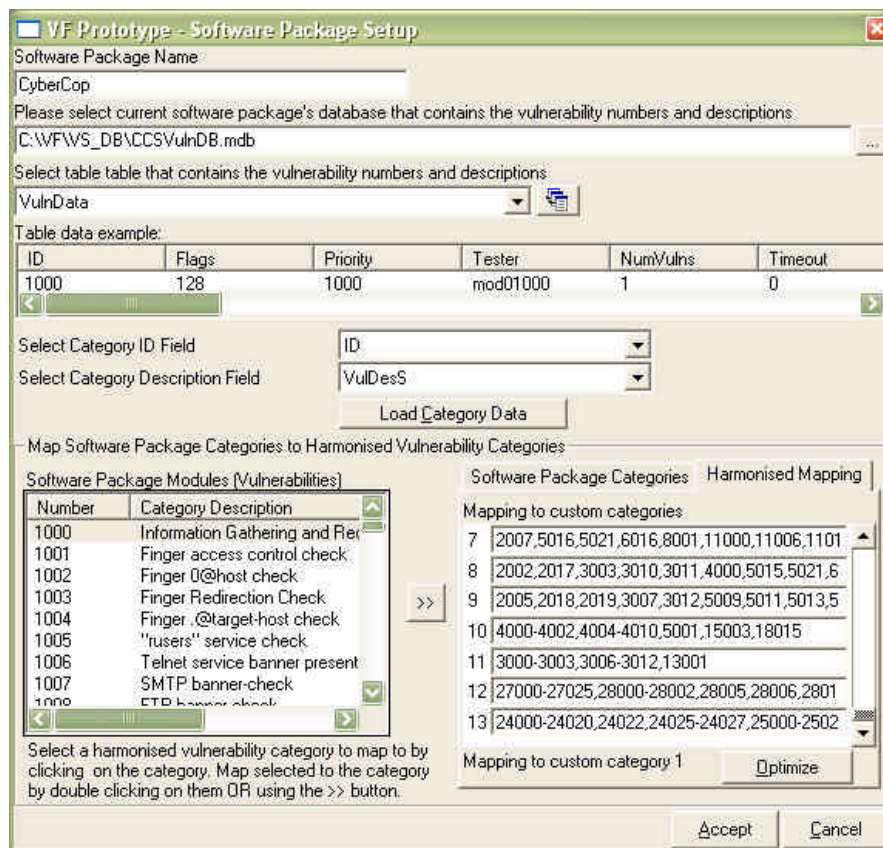


Figure 8.11: Software Package Setup window: Harmonised mapping

vulnerabilities. To indicate specifically to the VF Prototype which are the categories, all the software package modules that appear with an ID **Number** as a multiple of 1 000 should, therefore, be selected and moved over to the Software Package Categories on the right-hand side of the form using the >> button.


8.4.3.3.5 *Specifying the vulnerability mapping*

Once the 31 CyberCop Scanner vulnerability categories have been specified, the mapping of all ID **Numbers** that are **not** multiples of 1 000 should be mapped onto the harmonised vulnerability categories. This might be a tedious process and may take a fair amount of time, but it is a once-off process for the specific software package, e.g. CyberCop Scanner, since this mapping pattern will be saved and used each time a vulnerability forecast is done. The mapping onto the 13 harmonised vulnerability categories for CyberCop Scanner is shown in figure 8.11.

Only the mapping onto harmonised vulnerability categories 7 to 13 is visible in figure 8.11, but scrolling up in the **Harmonised Mapping** tab will reveal the mapping onto harmonised vulnerability categories 1 to 6. Note how the mapping format is done for each harmonised vulnerability category: the ID **Number** of the specific **Software Package Module** that should be mapped onto the particular harmonised vulnerability category is entered. Each ID **Number** is separated by a comma. Because it is possible for many sequential ID **Numbers** to be mapped onto a specific harmonised vulnerability category, such numbers can be optimised by clicking on the **Optimize** button. For example, consider harmonised vulnerability category 13: initially the ID **Numbers** 27001, 27002, ... , 27025 followed by the rest of the ID **Numbers** were entered in line 13. The ID **Numbers** 27001, 27002, ... , 27025 can be optimised by the VF Prototype, however, by only showing the first and the last number in this range, separated by a hyphen, i.e. 27001-27025.

As soon as the mapping numbers are entered, clicking on the **Accept** button as shown in figure 8.11 returns the user to the **VF Prototype – Options** window shown in figure 8.8.

8.4.3.3.6 Specifying the format of the history scan data

Back at the window in figure 8.8, click on the browse button , in the **Vulnerability Scanning Software Information** frame, to load the **Vulnerability Prototype – Set Up Names** window shown in figure 8.12. The goal of this window is to tell the VF Prototype what the format of the history scan data will look like.

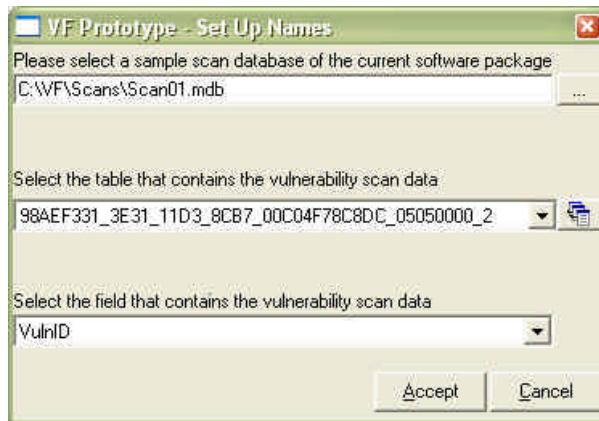



Figure 8.12: Specifying the scan data file structure of CyberCop Scanner

In the scan scenario with CyberCop Scanner, there are 15 databases available as history scan data – one database for each vulnerability scan that was conducted. The VF Prototype only needs an example of such a database to determine its format. In the first box of figure 8.12, enter the path to any one of these history scan databases, i.e. C:\VF\Scans\Scan01.mdb. Click on the **Load Tables** button  to load all the tables from the **Scan01.mdb** database to be populated into the drop-down list next to this button. The specific table with the desired history scan data for CyberCop Scanner is 98AEF331_3E31_11D3_8CB7_00C04F78C8DC_05050000_2. Likewise, the specific field containing the vulnerability ID is selected in the last drop-down box as VulnID. Click on the **Accept** button to return to the **VF Prototype – Options** window shown in figure 8.13.

Back at this window, click on **OK** to return to the main VF Prototype window as shown in figure 8.7. Note, however, that the **Software package used for vulnerability scan** box should now have the CyberCop entry selected.

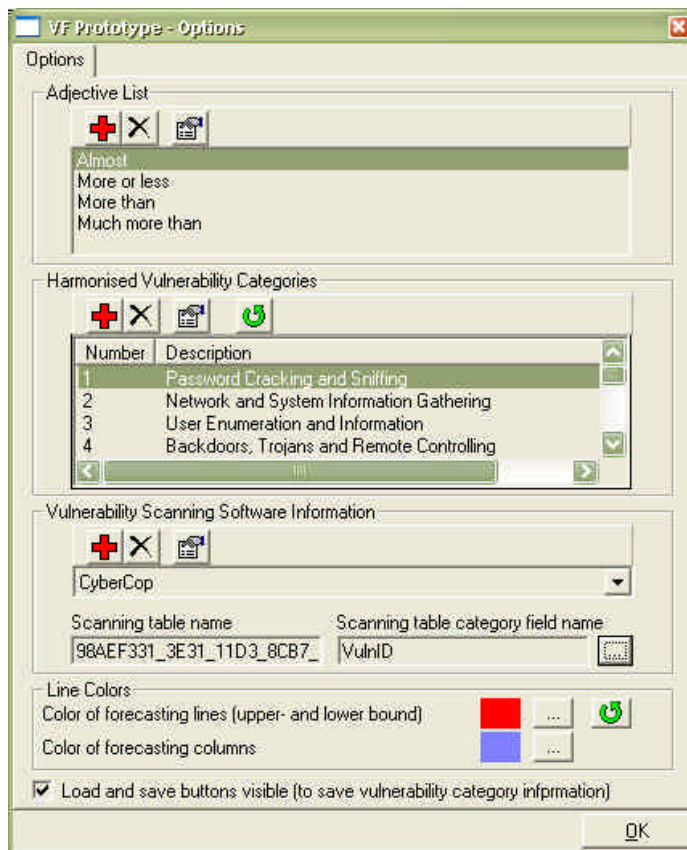


Figure 8.13: The VF Prototype – Options window filled in

Now that the initial setup procedure for the VF Prototype is complete, the prototype can start working on the history scan data.

8.4.4 Using the VF Prototype software

During this part of the VF Prototype software explanation, the history scan data is read and analysed by the prototype. A vulnerability forecast can then be made and its accuracy can be validated against an actual future scan.

8.4.4.1 Analysing the history scan data

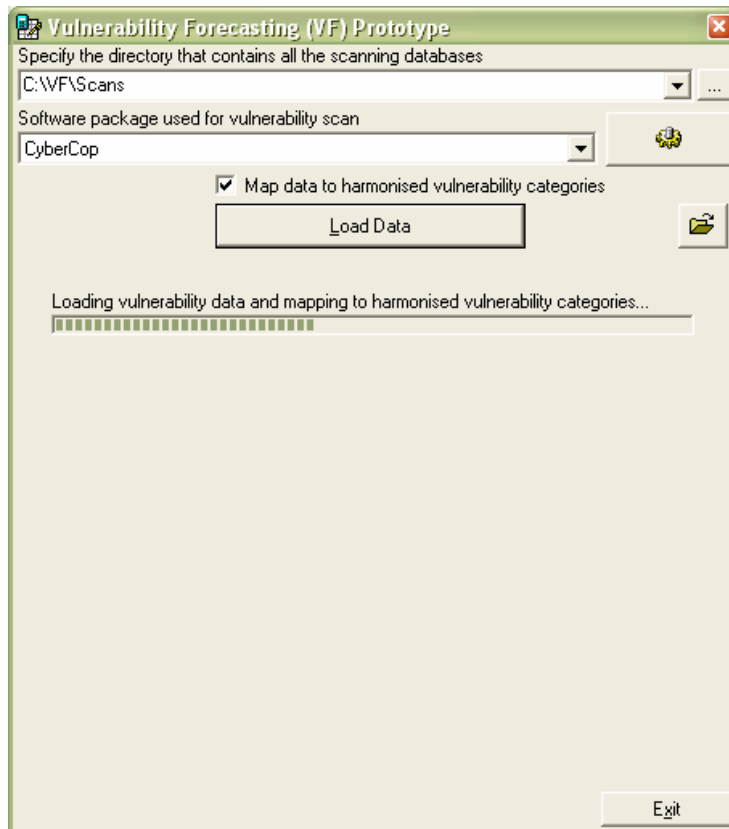


Figure 8.14: Loading the history scan data

At this stage, the history scan data is ready to be loaded by the VF Prototype in order to do a vulnerability forecast. In the main VF Prototype window, make sure that the **Map data to harmonised vulnerability categories** option is selected. When this option is not selected, no mapping of vulnerabilities onto the harmonised vulnerability categories is done, but the 31 vulnerability categories as defined by CyberCop Scanner will be used by the VF Prototype. The goal of mapping onto harmonised vulnerability categories, however, is to render the vulnerability forecast being made independent of specific VS products since the harmonised vulnerability categories act as a vulnerability “standard”. For the purposes of this scan scenario, however, mapping to the harmonised vulnerability categories is already done. Click on the **Load Data** button as shown in figure 8.14 to start loading the history scan data. This might take a minute or two, depending on the speed of the computer. After the data

has been loaded, the data for each of the 15 scans is shown under the **History Scan Data** tab. By double-clicking on a specific history scan database, i.e. **Scan15.mdb**, a graph for Scan 15 is shown, which indicates the number of vulnerabilities found for each harmonised vulnerability category. This graph is displayed in figure 8.16.

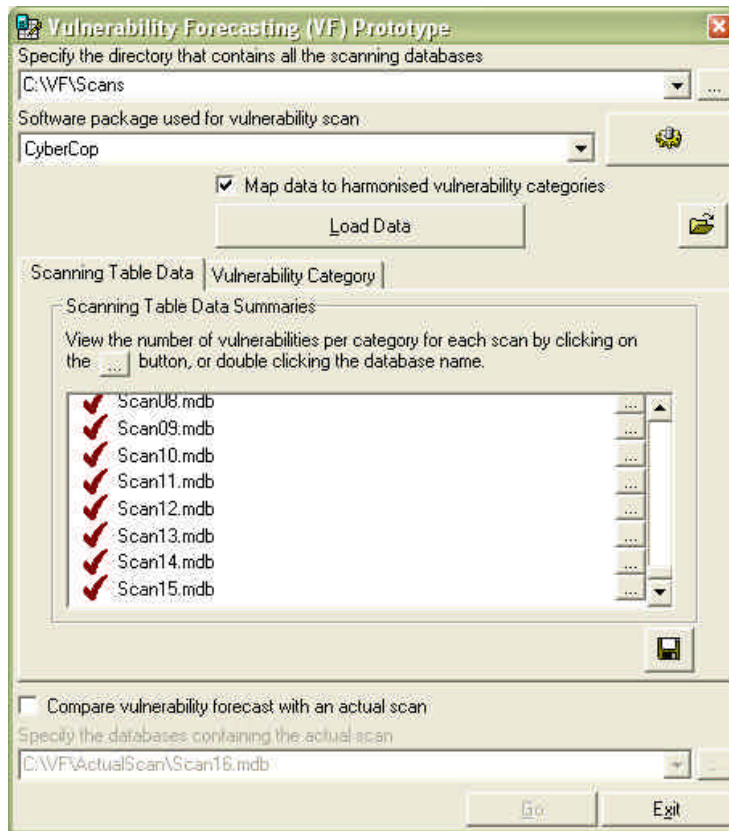


Figure 8.15: The history scan data loaded and mapped

JOHANNESBURG

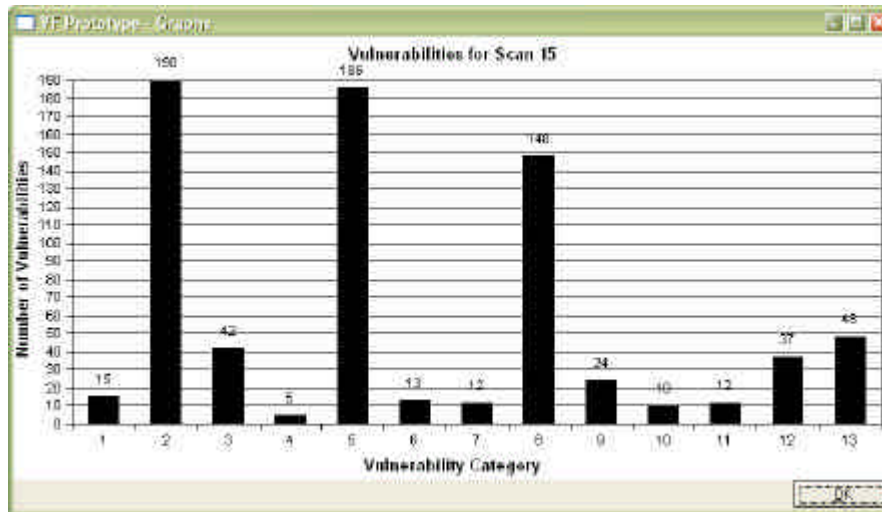


Figure 8.16: Graph showing information about Scan 15

8.4.4.2 Performing a vulnerability forecast

By clicking on the **Vulnerability Category** tab in figure 8.15, harmonised vulnerability category information will be displayed, as shown in figure 8.17. A vulnerability forecast can now be made for each of the harmonised vulnerability categories shown in figure 8.17. The five steps for doing a vulnerability forecast as explained in chapter 7 are applied and implemented in this part of the VF Prototype. These steps for doing a vulnerability forecast are the same for each harmonised vulnerability category. Therefore, only one harmonised vulnerability category – that of misconfigurations – will be discussed as an example.

Double-click on **Category 8** in the **Vulnerability Category Information** frame of figure 8.17. The **VF Prototype – Setup Information** window is displayed, as shown in figure 8.18. The **Mapping Table**, **Fuzzy Groups**, and the **Membership Function** are set up here for this specific harmonised vulnerability category. In the top right corner of figure 8.18, a miniature graph is shown, which displays the number of vulnerabilities found during each of the 15 history data scans for harmonised vulnerability category 8. Click on this miniature graph to enlarge it, as shown in figure 8.19.

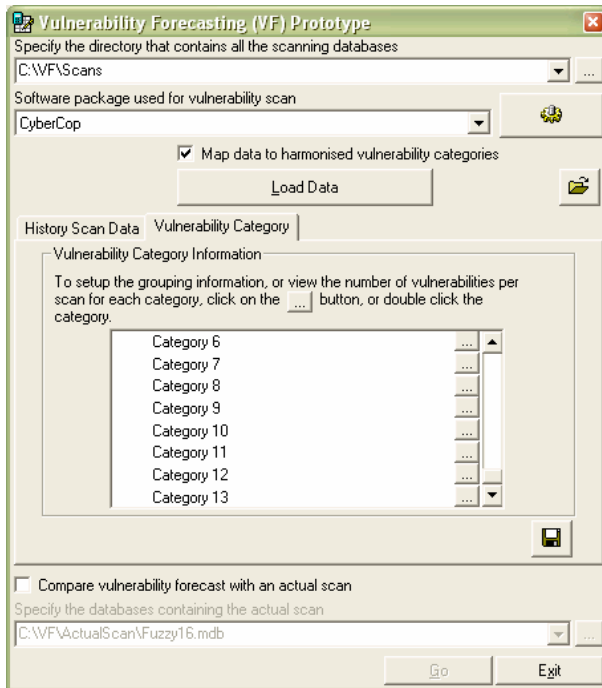


Figure 8.17: Showing the mapped data for the harmonised vulnerability categories

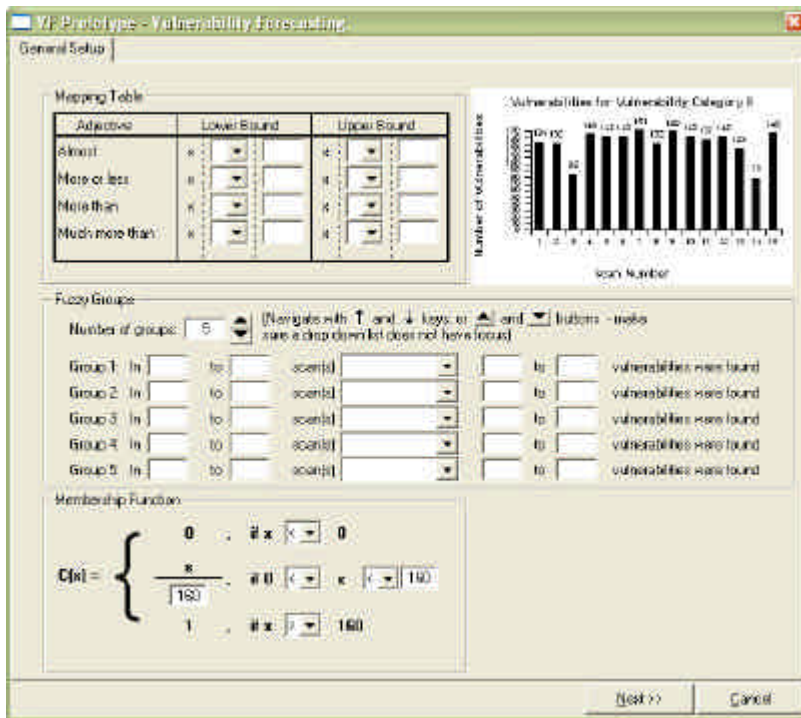


Figure 8.18: The first three steps for doing a vulnerability forecast for category 8

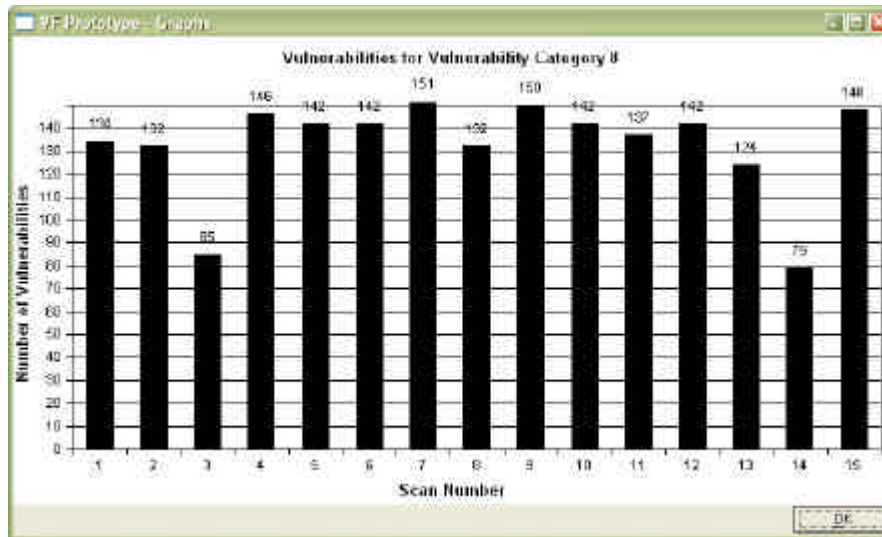


Figure 8.19: Graph showing information about harmonised vulnerability category 8

The graph displayed in figure 8.19 is specifically used by the user to fill in the window in figure 8.18. This graph is specifically used to compile the **Fuzzy Groups**, **Mapping Table**, and the **Membership Function**. The following three sections will explain how this is done for harmonised vulnerability category 8. Note, however, that these three sections correspond with steps 1 to 3 as explained in chapter 7 and input is required from the user during these three steps. During steps 4 and 5, however, the VF Prototype is able to calculate on its own.

8.4.4.2.1 Compiling the fuzzy groups

Determining the fuzzy groups is effected by examining the history data for harmonised vulnerability category 8, for example, over 15 history scans. Each of the 15 scans must be allocated to a specific fuzzy group. It is also possible for a scan to be allocated to more than one fuzzy group. The idea is to group together the harmonised history data of those scans in figure 8.19 that closely relate to one another.

Consider scans 3 and 14. Since these two scans revealed that their results are close to each other, they will be grouped into fuzzy group 1. Therefore, it can be said about fuzzy group 1 that *for exactly 2 vulnerability scans, the vulnerability range is exactly [79, 85]*. What exactly is meant by “close to each other”? In this context it means

that a fuzzy group normally comprises all scans that result within a range of 10 vulnerabilities from each other, for example the range between scan 3 and scan 14 for fuzzy group 1 is 7 (from 79 to 85). No other scan result is near this range, and therefore only these two scans are grouped together. The range of 10, therefore, is a suitable measure for this example to construct the fuzzy groups.

In the same way, scan 13 is the only scan that falls into a range between 120 and 130 and is the only scan that will be added to fuzzy group 2. Therefore, it can be said about fuzzy group 2 that *for exactly 1 vulnerability scan, the vulnerability range is exactly [124, 124]*, or simply 124.

The vulnerability scans grouped for the first two fuzzy groups were exact, in other words there was nothing “fuzzy” about them. Now consider scans 1, 2, and 8, which form fuzzy group 3. For all three of these scans, the **almost** reaches 135. In addition, scan 11 might also be included in this fuzzy group because it would fit well into a group range of 10. However, it could also fit in with fuzzy group 4, which will be discussed in the next paragraph. Therefore, it will be allowed for fuzzy group 3 to have between 3 and 4 scans included. For fuzzy group 3, therefore, it can be said that *[3, 4] vulnerability scans delivered a vulnerability range of almost [135, 135]*. The adjective here is considered to be “fuzzy”, because “almost” is not a clear-cut value.

The last fuzzy group for this example includes the remainder of the scans. Fuzzy group 4, therefore, consists of scans 1, 2, 4, 5, 6, 7, 9, 10, 11, 12, and 15. The distribution of the scan results in this fuzzy group constitutes **more or less** 145 vulnerabilities found. Following the same approach as stated in the paragraph above, however, scan 11 could either be included in this fuzzy group or in fuzzy group 3. Therefore, it can be said about fuzzy group 4 that *for [10, 11] vulnerability scans, the vulnerability range is more or less [145, 145]*. The adjective here is considered to be “fuzzy”, because “more or less” is not a clear-cut value.

The four fuzzy groups identified in this exercise are entered into the **Fuzzy Groups** frame in figure 8.18.

8.4.4.2.2 *Compiling the mapping table*

In the previous section, the adjectives “almost” and “more or less” were encountered. These are fuzzy values. A computer, unfortunately, does not know what “almost” and “more or less” mean, unless it is “clarified” by the mapping table.

Setting up the mapping table, however, requires a careful analysis of the vulnerability scan data in terms of each of the above adjectives. “Almost 135” referred to vulnerability scans 1, 2, and 8. The goal is to study these three scans in figure 8.19 to see exactly what something fuzzy, i.e. “almost”, means in terms of some crisp value range. This value range appears to be [132, 134] from the graph, where 132 was the smallest and 134 was the largest possible value delivered by the three scans. When one thinks logically about “almost”, it means that a value was never reached, although it came close. This logic, however, must now be applied to [132, 134]. The value 132, thus, came close to 135, but never reached it. In other words, the value came as close as 2% $(135-132)/135*100$ short to 135. In the same way, the value 134 came even closer to 135 – only 1% short. Therefore, the mapping table formula for representing the lower bound of “almost” x is $x - 2\%$, and the upper bound of “almost” is $x - 1\%$. Enter these values next to **Almost** in the mapping table.

Likewise, the data is studied in figure 8.19 for vulnerability scans 1, 2, 4, 5, 6, 7, 9, 10, 11, 12, and 15 to get a mapping table entry for “more or less” 145. The value range for these vulnerability scans appears to be [137, 151] from the graph. When one thinks logically about “more or less” 145, it means that a value was slightly less than 145, or it was slightly more than 145. This logic, however, must now be applied to [137, 151]. The value 137, thus, is 6% $(145-137)/145*100$ less than 145. In the same way, the value 151 is 4% more than 145. Therefore, the mapping table formula for representing the lower bound of “more or less” x is $x - 6\%$, and the upper bound of “almost” is $x + 4\%$. Enter these values next to **More or less** in the mapping table.

Any mapping table values for **More than** and **Much more than** can be entered in this demonstration, since they are not used for harmonised vulnerability category 8 in this example.

8.4.4.2.3 Compiling the membership function

The membership function simply indicates what the absolute maximum number of vulnerabilities will be according to the current history data for the current example. In this case, not one of the 15 scans ever showed more than 170 vulnerabilities found in harmonised vulnerability category 8. Enter 170 into both textboxes in the **Membership Function** frame.

Click on **OK** to dismiss the graph of figure 8.19. After the form in figure 8.18 is filled in, it should look like figure 8.20.

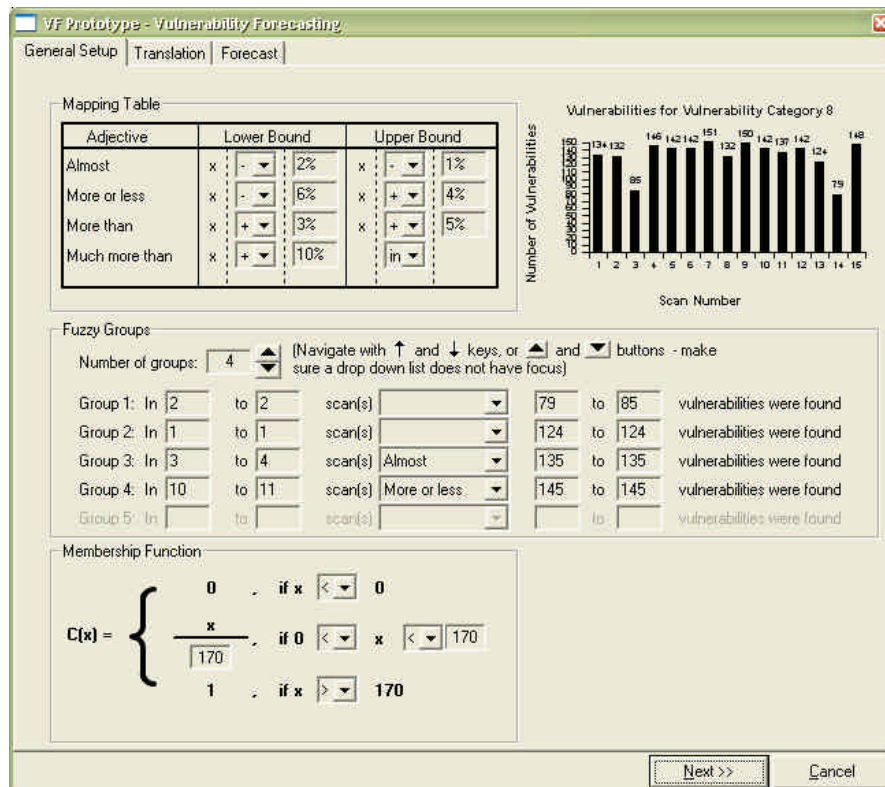



Figure 8.20: The first three steps for doing a vulnerability forecast in the VF Prototype completed

8.4.4.2.4 Viewing calculations

In order to complete step 4 and view the calculations of how the values up to step 4 have been constructed, click on the **Translation** tab. A window with all the calculations of values up to step 4 is shown in figure 8.21. The detailed calculations of the m -values can be viewed by clicking on one of the **View Calculations** buttons . The calculations for calculating the third m -value are shown in figure 8.22.

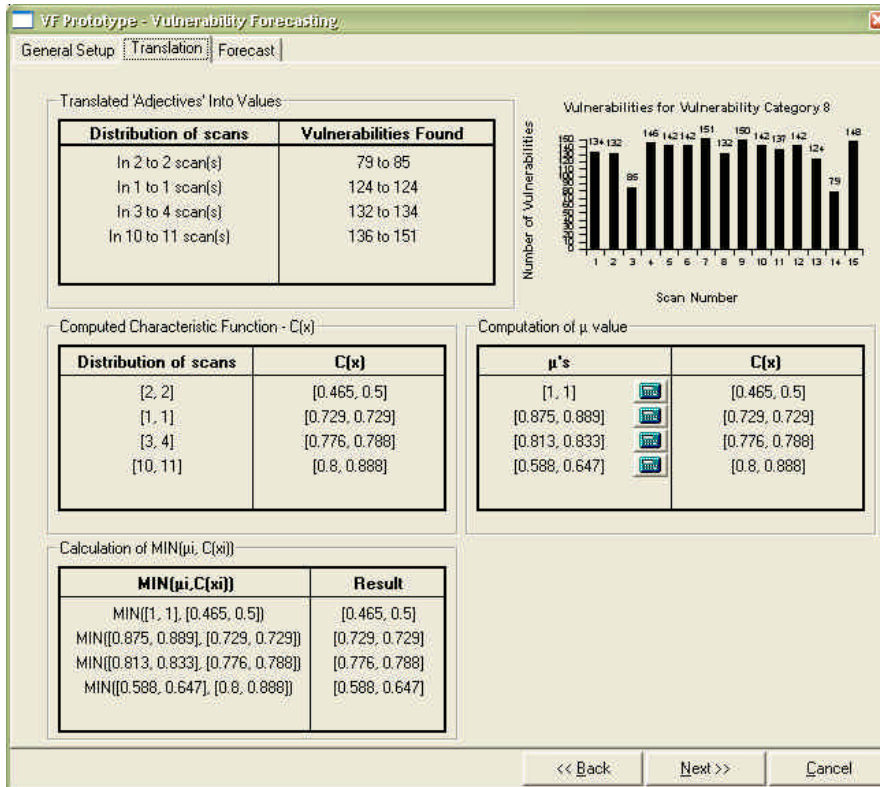


Figure 8.21: Step 4 for doing a vulnerability forecast in the VF Prototype

Click on **OK** to dismiss the **Calculations** window. Step 5 and the final vulnerability forecast result is shown in the **Forecast** tab of figure 21. Clicking on this tab opens the window shown in figure 8.23.

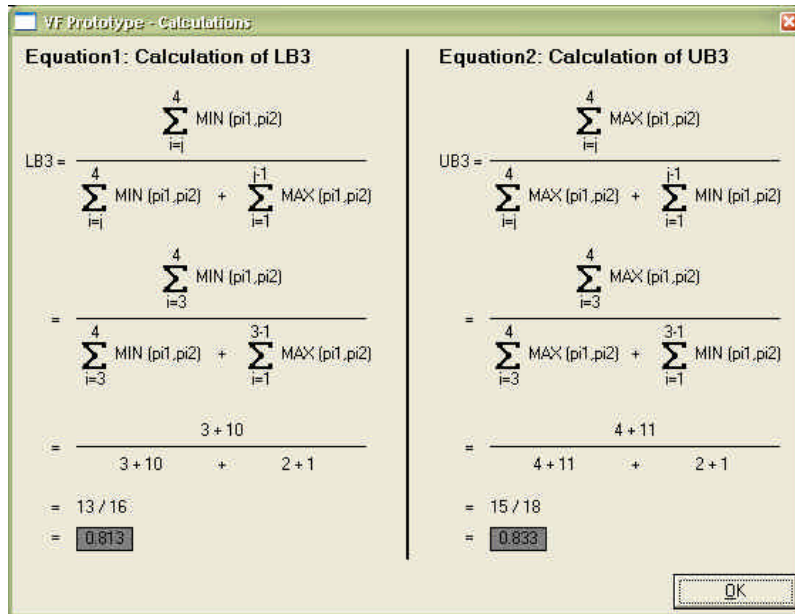


Figure 8.22: Calculations for the fourth m-value as displayed in figure 8.21

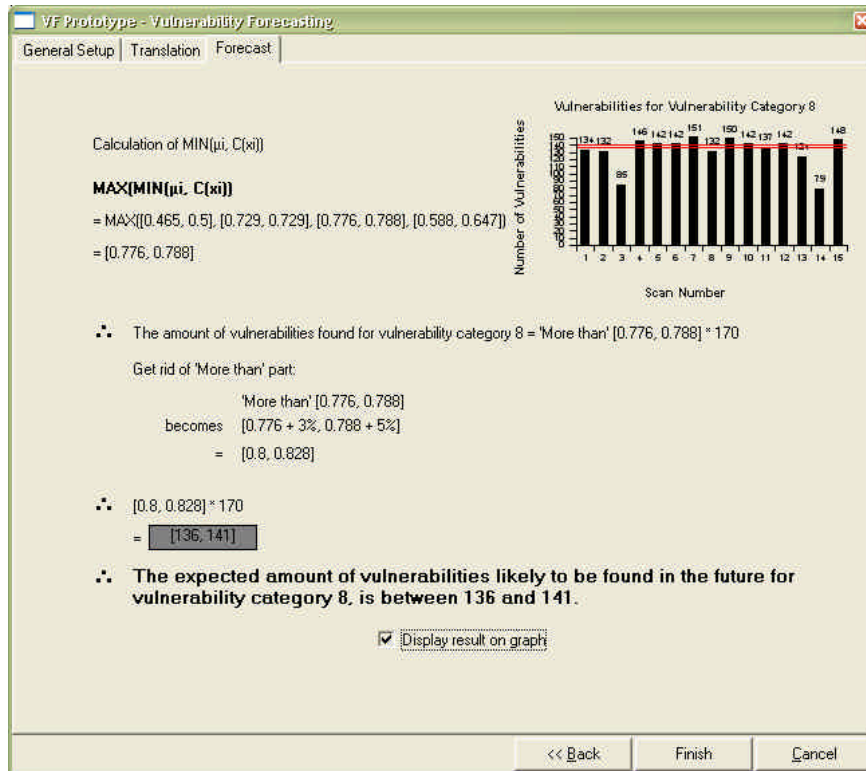


Figure 8.23: Step 5 and the final vulnerability forecast result for harmonised vulnerability category 8

The final vulnerability forecast result for harmonised vulnerability category 8 states that, by the next time a vulnerability scan, i.e. Scan 16, is conducted, it is forecast that between 136 and 147 misconfiguration vulnerabilities will be detected by CyberCop Scanner. By selecting the **Display result on graph** option, this forecast range is also indicated by the red lines in the graph in the top right corner of figure 8.23. Once again, this graph can be enlarged by clicking on it, and is shown in figure 8.24. Click on **OK** to dismiss the graph. Likewise, the vulnerability forecasts for each of the other harmonised vulnerability categories can be obtained using the VF Prototype. Click on **Finish** to dismiss the **VF Prototype – Setup Information** window.

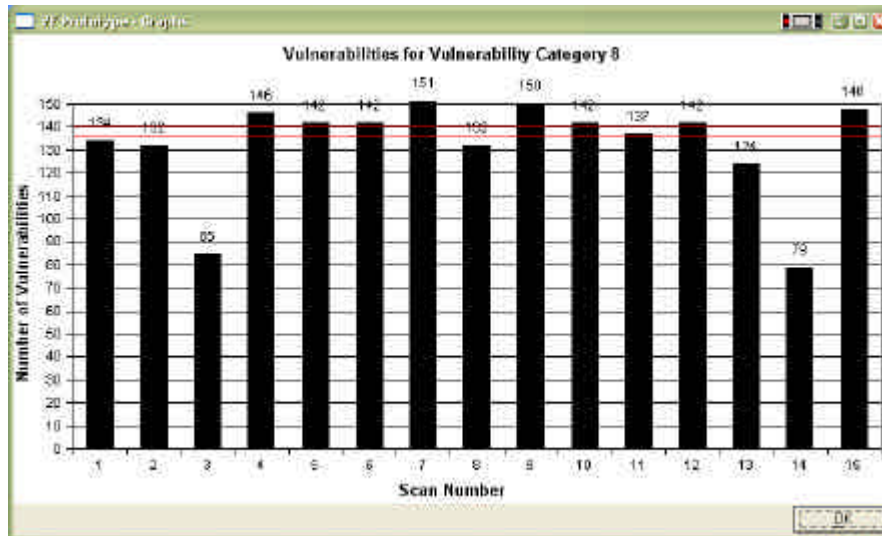


Figure 8.24: Graph showing information about harmonised vulnerability category 8

A red tick mark should appear next to harmonised vulnerability category 8 to indicate that a vulnerability forecast for this harmonised vulnerability category has been done. The main VF Prototype window therefore should look like figure 8.25 when vulnerability forecasts for all harmonised vulnerability categories have been done.

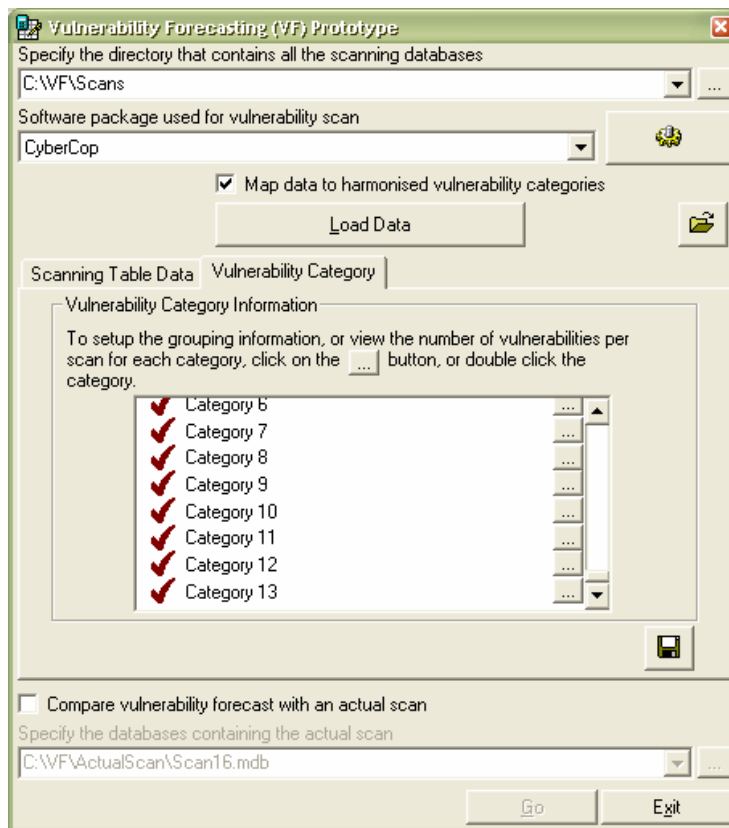


Figure 8.25: The completed VF Prototype main window

The next section will demonstrate how these vulnerability forecasts can be validated.

8.4.4.3 Validating a vulnerability forecast

In order to see the accuracy of the vulnerability forecasts, they can be compared with an actual scan. In other words, a 16th vulnerability scan was conducted so that the forecasting results can be compared with it. Figure 8.26 shows a comparison of an actual scan – Scan 16 – and the vulnerability forecast. Note that the scan interval between Scan 15 and Scan 16 was one day, which was the same between all the other 15 scans.

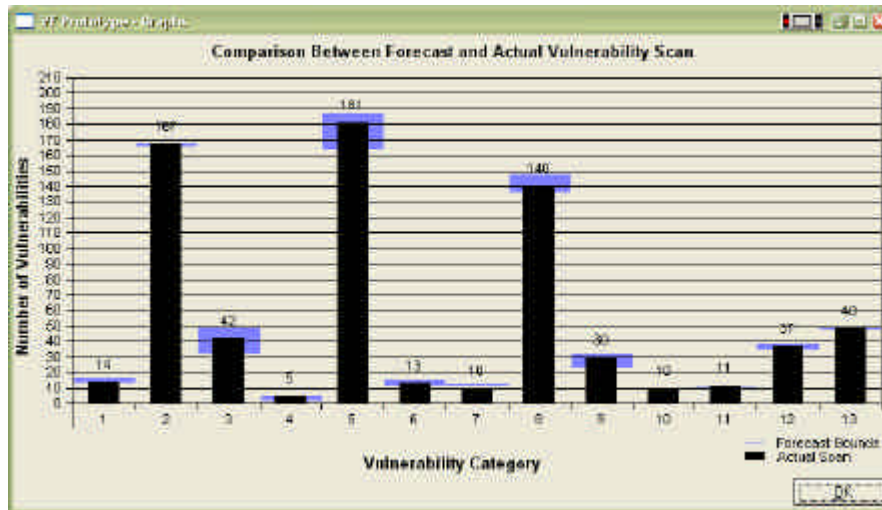


Figure 8.26: Comparing a vulnerability forecast with an actual scan – Scan 16

The black bars in figure 8.26 denote Scan 16 while the blue blocks indicate the vulnerability forecast range for each harmonised vulnerability category. The results of the vulnerability forecast compared to the actual scan are 92% accurate because the vulnerability occurrences for 12 of the 13 harmonised vulnerability categories fell precisely into the forecast vulnerability ranges. The only misforecast harmonised vulnerability category turned out to be category 7, but this vulnerability forecast was not that far out anyway.

The accuracy of the vulnerability forecast was so high because vulnerability scans 1 to 16 were conducted at regular one-day intervals. If the time interval between Scan 16 and a next scan is significantly increased by a 45-day interval, however, the vulnerability forecast is not far out, as shown in figure 8.27. This result, therefore, proves that, by significantly increasing the time interval between scans, the vulnerability forecast still gives a very good indication of how vulnerabilities will occur in the future.

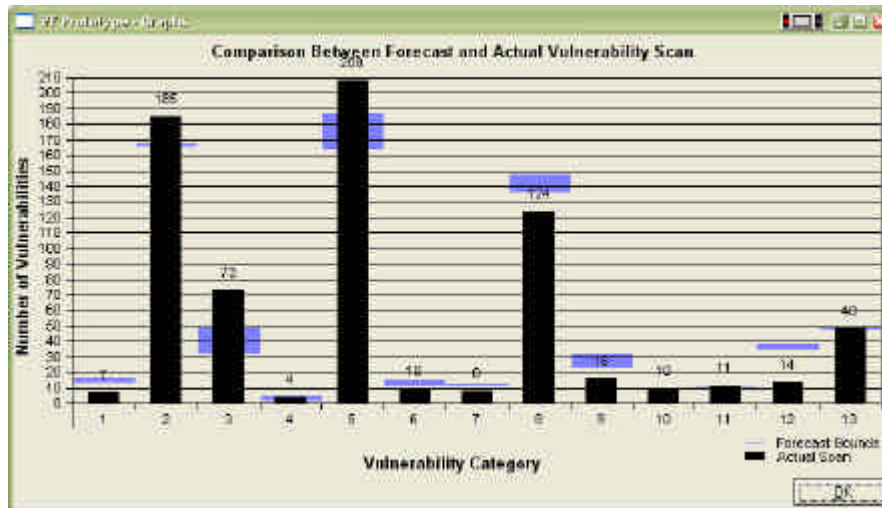


Figure 8.27: Comparing a vulnerability forecast with an actual scan for a significant time interval increase of 45 days between scans instead of every day

8.5 CONCLUSION

This chapter demonstrated how vulnerability forecasting can be implemented in a prototype. The prototype also assisted in demonstrating how some of the desired features for vulnerability forecasting can be attained.

The main conclusion about the VF Prototype is that it clearly shows that increasing the time intervals between successive vulnerability scans results in fewer vulnerability scans having to be conducted. This is a major improvement in working with VS products, because fewer scans having to be conducted means that less system resources and time will be occupied by the vulnerability scanner in the future. In addition, the VF Prototype shows that vulnerability scans need not be conducted at regular time intervals as long as enough history scans are available, because whether the scan interval between vulnerability scans is 1 day or 45 days, as in this test scenario, the vulnerability forecast proves not to be far out.

This research culminates at this stage of the thesis and the specific problems with VS products as identified in chapter 6 are revisited in table 8.3, which indicates that the VF Prototype has indeed addressed the selected problems.

Table 8.3: Problems identified and addressed regarding state-of-the-art VS products

Problems identified	Problem addressed	Remark
1. Conducting vulnerability scans is too time-consuming.	✓	Vulnerability scans are still time-consuming, but having to conduct fewer scans means that less time will be consumed in retrospect.
2. A VS product generally occupies a vast number of network and system resources, leading to the degradation of system performance while vulnerability scans are being conducted.	✓	VS products are still occupying a vast number of network and system resources while scanning, but having to conduct fewer scans means that less of these resources will be occupied in retrospect.
3. VS products lack intelligence because they are unable to learn about new vulnerabilities automatically.	✓	Fuzzy logic was implemented in the VF Prototype to improve the utilisation of harmonised vulnerability categories.
4. The vulnerability database structure differs extensively from one VS product to another.	✓	A standardised relational database has been created in order to store and manage vulnerabilities in a harmonised manner.
5. The types of vulnerabilities being scanned for differ extensively from one VS product to another.	✓	Using harmonised vulnerability categories "harmonised" the vulnerabilities of different VS products and hence such vulnerabilities are viewed as a "standard".
6. Scans may not always be conducted at regular intervals due to unforeseen circumstances, for example when critical maintenance on servers and the network is carried out.	✓	It was proven that scans can be conducted at irregular time intervals with very little difference in the vulnerability forecast outcomes.
7. The vulnerability database should be updated before a scan is conducted, otherwise the scan result may not be accurate enough.		Not implemented.
8. Most rectification procedures cannot be automated and still require the expertise of qualified personnel.		Not implemented.
9. VS products do not provide adequate and sufficient information that would aid risk management.	✓	Having a vulnerability forecast would aid in risk management of vulnerabilities in the future.

The next chapter will conclude with the final remarks about the VF model, the VF Prototype implementation, and future work.

CHAPTER 9

CONCLUSION

9.1 INTRODUCTION

This thesis represented the model for vulnerability forecasting (VF) – a model that enhances security in the domain of networks and the Internet. In this chapter, the researcher will evaluate the extent to which the objectives of this research study have been achieved. Finally, the researcher will conclude the thesis with further research suggested and an epilogue.

9.2 REVISITING THE PROBLEM STATEMENT

The principal aim of this research study was to make a contribution to proactive information security technologies, i.e. vulnerability scanning, in the Internet and network security domain. To do so, the researcher set out to develop a model for VF that would be specifically tailored to this domain.

The problems that were to be addressed in the study, as stated in chapter 1 in the form of research questions, will be re-examined in the sections that follow with a view to ascertaining the extent to which this had been accomplished.

What is the state of current proactive and reactive information security technologies?

A taxonomy for state-of-the-art information security technologies was introduced in chapter 2. The two mainline information security technologies included *proactive* and *reactive* information security technologies. Each of the proactive and reactive information security technologies was subdivided into network-, host- and application-level information security technologies.

A number of resources were identified and the extent to which the information security technologies were covered in those resources was indicated. It seems that these technologies are covered more extensively in journals than in books.

Chapter 3 investigated a reactive information security technology – intrusion detection – and a proactive information security technology – vulnerability scanning. The architectures of both these technologies were investigated extensively and, after considering a number of problems of both, it was decided to select vulnerability scanning as the information security technology on which the remainder of the thesis would be based.

State-of-the-art proactive and reactive information security technologies, therefore, form the enablers to a wide range of information security products available on the security software market today. In addition, they serve as the building blocks to newer and better information security implementations.

What can be done to improve the vulnerability scanning process?

Since the model for VF, as proposed in this thesis, utilises existing vulnerability scanners to conduct vulnerability scans, no improvement in vulnerability scanning products themselves has been suggested.

Instead, however, the vulnerability scanning process has been improved, as shown in chapter 6, in that fewer vulnerability scans are conducted when vulnerability forecasts are made. In addition, this also buys more time and, therefore, lessens the administrative overhead of working through lengthy vulnerability scanner reports to rectify vulnerabilities.

How can the impact of current vulnerability scanners on system resources be minimised?

Chapter 5 provided a detailed study of specific vulnerability scanner products and reported on the researcher's practical experience and the database structures of current vulnerability scanner products. It was explained in chapter 3 that the modus operandi of a vulnerability scanner is to simulate attacks to test whether the system resources

have been secured sufficiently. There is no possibility, unfortunately, of changing this modus operandi of current vulnerability scanners.

An alternative scanning strategy has, however, been suggested and discussed in chapters 6 and 7 – that of being able to conduct fewer vulnerability scans without compromising the detection of vulnerabilities, and being able to forecast which vulnerabilities will occur in the mean time before the next vulnerability scan is conducted. The increased time frame between successive vulnerability scans, therefore, means that the abundant consumption of system resources will occur less frequently. The impact of current vulnerability scanners on system resources is, therefore, minimised.

How can the disparity be addressed in the kinds of vulnerabilities that different vulnerability scanner products can detect?

The disparity problem between different vulnerability scanners was addressed in chapter 3. A solution to this problem was suggested in chapter 4, where the concept of harmonised vulnerability categories was introduced.

The harmonised vulnerability categories provide a standard method for grouping related vulnerabilities and, thus, enable one to know, which subset of standardised vulnerabilities a specific vulnerability scanner can detect from a potentially exhaustive set of standardised vulnerabilities.

How should vulnerability scanner products provide more intelligent results so that they will aid risk management?

Chapter 6 identified some specific problems with current vulnerability scanner products, after which the concept of vulnerability forecasting was introduced. The model for VF has been explained in detail, but chapter 7 was specifically devoted to the heart of the VF model, namely the vulnerability forecast engine. It is in the latter chapter that “more intelligent results” have been provided by the model for vulnerability forecasting. The VF model was tested by a prototype as described in chapter 8.

Looking at the results of chapter 8, a vulnerability forecast is provided, which would enable management to analyse the risk of the vulnerability forecast. In doing so, vulnerability forecasting more intelligently provides human resources with the ability to effect risk management.

9.3 FUTURE RESEARCH

The proposed model achieved the objectives to the extent described in the section above, but it suffers from some limitations. The limitations, fortunately, provide opportunities to extend and support the work described in this thesis by a number of future research projects as presented below:

- Further research is possible in a bid to find techniques on how to automate the procedure of mapping vulnerabilities of current VS products onto the harmonised vulnerability categories, because this process is done manually in the current research project. In addition, when the specific VS product's vulnerability database is updated, the new vulnerabilities are currently mapped manually onto the harmonised vulnerability categories.
- Having said the above, another potential research project can be identified: integrating current VS technology and vulnerability forecasting in a single vulnerability-forecasting-enabled VS. In this way the administration of vulnerability forecasting will be decreased even further.
- The process of configuring the vulnerability forecast engine according to the history scan data before a vulnerability forecast can be made is also currently done manually. Automating this process requires more intelligent techniques such as using the graphical information created from the history scan data to automatically set up the fuzzy groups, creating the mapping table and defining the membership function. The administration of vulnerability forecasting will yet again be decreased by such intelligent techniques.
- For the current research project, the researcher claimed that having *sufficient* history data would enable one to do a vulnerability forecast. However, on what grounds can one claim that the history data gathered is *sufficient*? In addition, for *how long* will a forecast remain valid? These time-frame-based questions are questions that may be addressed in a research project of its own merit.

- Vulnerability forecasting is built upon the specific information security technology called vulnerability scanning. It might be possible, as a future research project, to combine vulnerability scanning with other information security technologies such as intrusion detection systems and firewalls. In this way, hybrid vulnerability forecasting might be possible.
- Finally, one might ask whether there may be other means of implementing intelligent techniques to do a vulnerability forecast. The technique used in this research project – fuzzy logic – might be replaced or enhanced by using techniques such as neural networks [HAMB 93] or fuzzy cognitive maps [YELZ 95].

9.4 EPILOGUE

The researcher concludes this research project with the following quote:

“Research is what you’re doing, when you don’t know what you’re doing.”

Charles Wilson, president of General Electric, 1945-1950

This quote may prove to make sense at the start of a research project, but, is it not wonderful in the world of research that, the closer the research project progresses to the end, the less this quote makes sense? This has been the experience of the researcher throughout the completion of this research project. The researcher hopes that the work presented in this thesis will stimulate further research – not only in the particular future research projects mentioned above, but also in the entire domain of Internet and network security. The researcher believes that this thesis is able to do that, and, to prove this, leaves its readers with the following truth about research:

“To research there is no end, only new beginnings.”

*Reinhardt A. Botha, CoSAWoE – A Model for Context-sensitive
Access Control in Workflow Environments, Chapter 12*



APPENDIX A

INSTALLING THE VF PROTOTYPE SOFTWARE AND ADDITIONAL SOFTWARE COMPONENTS

Appendix A serves as a guide to installing and getting started with the VF Prototype.

A.1 INSTALLATION REQUIREMENTS

The following minimum system requirements would have to be met in order to run the VF Prototype:

- A Pentium III 500 MHz or equivalent processor.
- 64 MB RAM.
- Microsoft® Windows 98/2000/XP.

It is recommended, however, that the prototype be executed on a computer with 128 MB RAM or more to ensure better performance. In addition, an 800 x 600 pixels or more Super VGA display screen would enable a clearer display of graphs.

A.2 CONVENTIONS USED

To explain, in special cases, how the VF Prototype works, the prototype employs the typefaces described in table A.1 to indicate special text.

Table A.1 : Typefaces employed to indicate special text

Typeface	Meaning
Courier New typeface	Courier New text represents text as it appears on the screen or as it should be entered into a field on the screen.
<i>Italics</i>	Italicised text represents commands, for example the text <i>Start</i> indicates that the “Start” button in Windows should be activated.
→	This symbol is used as a separator that separates commands. To open the “Run” command in Windows, for example, the text sequence <i>Start</i> → <i>Run</i> indicates that the “Start” command should be activated, followed by the “Run” command.
Boldface	Boldface text is used to emphasise certain terms and to refer to certain objects, buttons or labels in the VF Prototype.

Note that the terms “directory” and “folder” are used interchangeably and have the same meaning in this context.

A.3 INSTALLING THE VF PROTOTYPE SOFTWARE

The VF Prototype software can be downloaded from the Web at the following URL: <http://www.cs.up.ac.za/~hventer>. Once at this web site, the following steps can be taken to download and install the software:

- Downloading the VF Prototype software:
 - Create a temporary directory on the hard drive of your computer and call it, for example, **C:\temp**.
 - Point your Web browser to the following URL: **<http://www.cs.up.ac.za/~hventer/vf/vf.exe>**.
 - Download the **VF.exe** installation file onto the hard drive of your computer into the **C:\temp** directory you have created.
- Installing the VF Prototype software:
 - Click on *Start* → *Run* and enter **C:\temp\VF.exe** into the **Open** field as shown by the **Run** window in figure A.1.

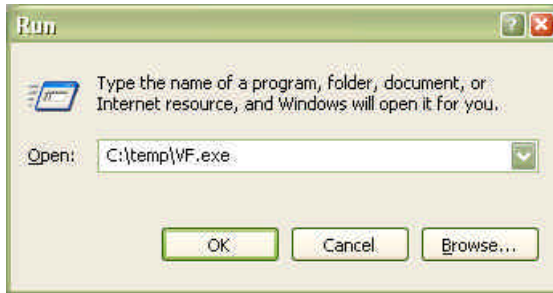


Figure A.1: Entering the command to run the VF installation software

- Click on the **OK** button. The VF installation software should open as shown in figure A.2.
- Change the text in the **Destination folder** field to C:\. Note that the installation software [ROSH 03] will automatically create the C:\VF subdirectory on your computer's hard drive. Strictly C:\ only should be entered in the **Destination folder** field.
- Click on the **Install** button to start the installation process. Figure A.3 shows the installation progress of the VF Prototype.

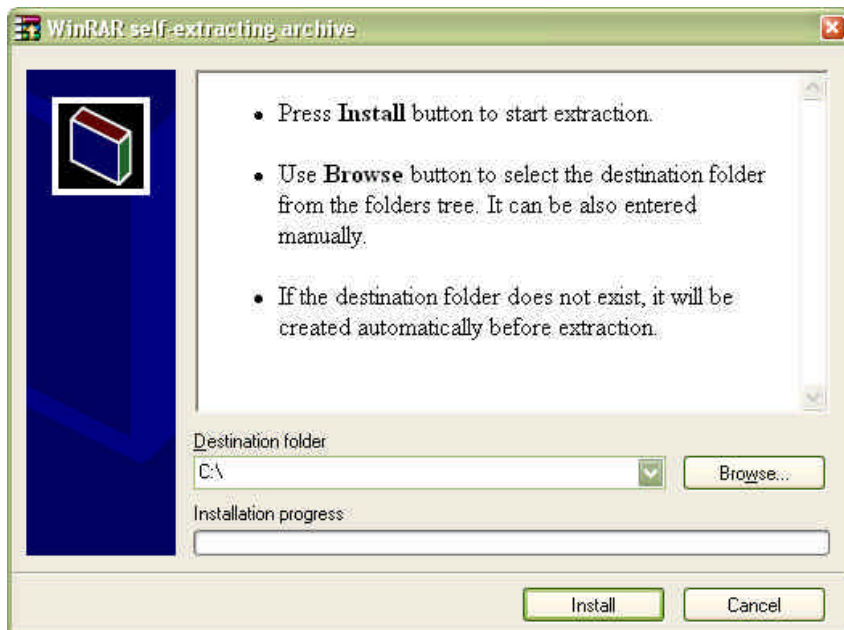


Figure A.2: Specifying the destination folder for installing the VF Prototype software

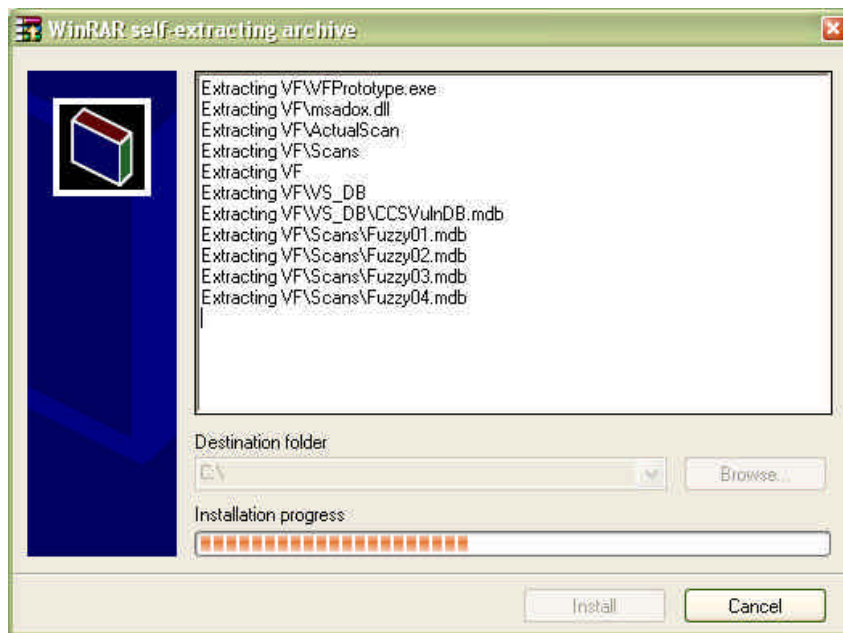


Figure A.3: The VF Prototype files being installed by the installation software

The VF Prototype software installation procedure is not complete at this stage. Some additional software components still need to be installed before the VF Prototype software will run correctly. The next section provides the installation procedure for these additional software components.

A.4 INSTALLING ADDITIONAL SOFTWARE COMPONENTS

The VF Prototype uses additional components, referred to as ActiveX controls [MICR 03], to run successfully. An ActiveX control is a software component that was developed for a general purpose, e.g. to add additional programming functionality. Two ActiveX controls were used in the VF Prototype and should be installed as follows:

- The **COMDLG32.OCX** file should already be extracted and available in the C:\VF directory that was created with the VF Prototype software installation process.
- The **COMDLG32.OCX** file needs to be registered in the Windows registry. In order to register this file, click *Start* → *Run* in Windows.

- Type the following register command `regsvr32 C:\VF\COMDLG32.OCX` in the **Open** field of the **Run** window as shown in figure A.4.

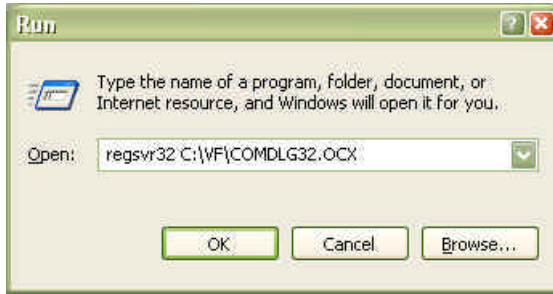


Figure A.4: Running the register command

- Click on the **OK** button in the **Run** window. The message box shown in figure A.5 should appear if the registration process was successful.



Figure A.5: Successful component registration

- Click on the **OK** button to dismiss the message box in figure A.5.
- Repeat all the steps of this section for installing additional software components, but do so for the file **MSCOMCTL.OCX** instead of **COMDLG32.OCX**. The file **MSCOMCTL.OCX** should also be available in the **C:\VF** directory after installation of the VF Prototype software.

The VF Prototype software should now be installed successfully and is ready to run.



APPENDIX B

SOURCE CODE OF THE VF PROTOTYPE

The source code for the VF Prototype is written in Microsoft Visual Basic 6.0 and is provided in this appendix. A screenshot of the project layout as displayed in visual basic is shown in figure B.1.

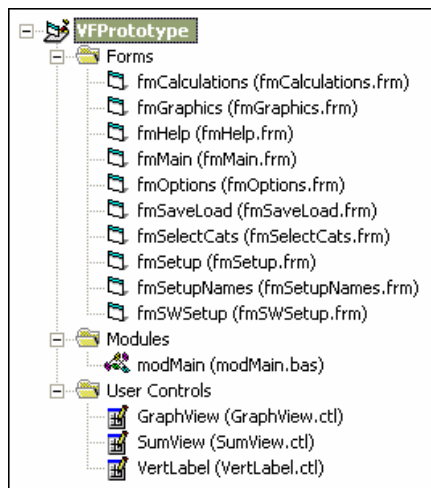


Figure B.1: Project layout of the VF Prototype

The source code is presented in the order of the forms, module, and user controls listed in figure B.1. A functional description is also provided for each of these.

B.1 SOURCE CODE FOR THE FORMS

B.1.1 The “frmCalculations” form

The design of this form is shown in figure B.2. This form is used to show some of the detailed steps in calculating the fuzzy expected value.

Equation1: Calculation of LBJ

$$LBJ = \frac{\sum_{i=1}^n \text{MIN}(pi1, pi2)}{\sum_{i=1}^n \text{MIN}(pi1, pi2) + \sum_{i=1}^{i-1} \text{MAX}(pi1, pi2)}$$

$$= \frac{\sum_{i=1}^4 \text{MIN}(pi1, pi2)}{\sum_{i=1}^4 \text{MIN}(pi1, pi2) + \sum_{i=1}^{1-1} \text{MAX}(pi1, pi2)}$$

$$= \frac{\text{Top}}{\text{Bottom_left} + \text{Bottom_right}}$$

$$= \text{Top/Bottom}$$

Equation2: Calculation of UBj

$$LBJ = \frac{\sum_{i=1}^n \text{MAX}(pi1, pi2)}{\sum_{i=1}^n \text{MAX}(pi1, pi2) + \sum_{i=1}^{i-1} \text{MIN}(pi1, pi2)}$$

$$= \frac{\sum_{i=1}^4 \text{MAX}(pi1, pi2)}{\sum_{i=1}^4 \text{MAX}(pi1, pi2) + \sum_{i=1}^{1-1} \text{MIN}(pi1, pi2)}$$

$$= \frac{\text{Top}}{\text{Bottom_left} + \text{Bottom_right}}$$

$$= \text{Top/Bottom}$$

Result

OK

Figure B.2: The “frmCalculations” form

The source code for this form follows below.

```
Option Explicit

Private Sub cmdOK_Click()
    Unload Me
End Sub

Private Sub Form_Load()
    'Center window on screen
    Me.Left = (Screen.Width - Me.Width) / 2
    Me.Top = (Screen.Height - Me.Height) / 2

    gbFmCalculationsLoaded = True
End Sub

Private Sub Form_Unload(Cancel As Integer)
    gbFmCalculationsLoaded = False
End Sub
```

B.1.2 The “frmGraphics” form

The design of this form is shown in figure B.3. This form is used to display the graphs in the VF Prototype.

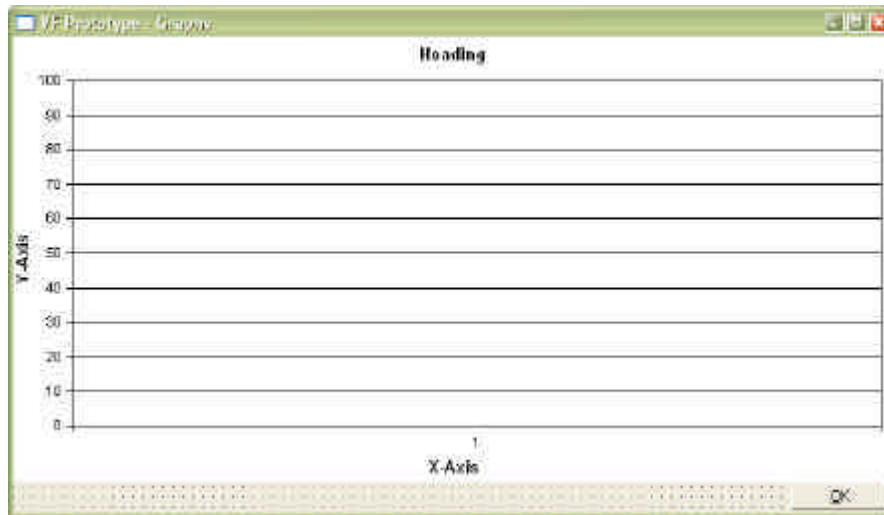


Figure B.3: The “frmGraphics” form

The source code for this form follows below.

```
Option Explicit

Private Sub cmdOK_Click()
    Unload Me
End Sub

Private Sub Form_Activate()
    Screen.MousePointer = vbNormal
End Sub

Private Sub Form_Load()

    'Center form on screen
    Me.Left = (Screen.Width - Me.Width) / 2
    Me.Top = (Screen.Height - Me.Height) / 2

    cmdOK.Visible = True

    gbFmGraphicsLoaded = True

End Sub

Private Sub Form_Resize()

    If Me.WindowState <> vbMinimized Then
        GraphView1.Move 0, 0, Me.ScaleWidth, Me.ScaleHeight - cmdOK.Height - 40
        cmdOK.Move Me.ScaleWidth - cmdOK.Width, GraphView1.Height + 10
    End If

End Sub

Private Sub Form_Unload(Cancel As Integer)
    gbFmGraphicsLoaded = False
End Sub
```

B.1.3 The “frmHelp” form

The design of this form is shown in figure B.4. This form is used to display information about the items in the vulnerability database of the particular VS product used by the VF Prototype, e.g. descriptions of vulnerability categories and vulnerabilities.

Figure B.4: The “frmHelp” form

The source code for this form follows below. In order to enhance the readability and to distinguish between the procedures, a horizontal line separates each procedure.

```
Option Explicit

Dim bIsActive As Boolean
Dim bExpanded As Boolean

'Constants
Private Const vSYSTEM_INTEGRITY As Long = 1
```

```

Private Const vCONFIDENTIALITY As Long = 2
Private Const vACCOUNTABILITY As Long = 4
Private Const vDATA_INTEGRITY As Long = 8
Private Const vAUTHORIZATION As Long = 16
Private Const vAVAILABILITY As Long = 32
Private Const vINTELLIGENCE As Long = 64

Private Type VulnType
    ID As Long
    Name As String

    RiskFactor As Long
    Complexity As Long
    Fixease As Long
    Popularity As Long
    RootCause As Long
    Impact As Long
    VulDesV As String
    Suggestion As String
    ManagerDesc As String
End Type

Private Type CatsType
    ID As Long
    Name As String
    Vulnerabilities() As VulnType
End Type
Dim Cats() As CatsType

```

```

Private Sub cmdMore_Click()

    Dim lstItm As ListItem
    Dim Count As Long

    If bExpanded Then
        txtVDescription.TabStop = False
        txtSuggestion.TabStop = False
        txtManDescription.TabStop = False

        Me.Height = 3960
        cmdMore.Caption = "&More Info >>"
        bExpanded = False
    Else
        bExpanded = True
        For Count = 1 To lvVulnerabilities.ListItems.Count
            If lvVulnerabilities.ListItems(Count).Selected Then
                Set lstItm = lvVulnerabilities.ListItems(Count)
                Call lvVulnerabilities_ItemClick(lstItm)
                Exit For
            End If
        Next Count

        txtVDescription.TabStop = True
        txtSuggestion.TabStop = True
        txtManDescription.TabStop = True
        Me.Height = 7905
        cmdMore.Caption = "&More Info <<"
    End If

End Sub

```

```

Private Sub cmdOK_Click()
    Unload Me
End Sub

```

```

Private Sub Form_Activate()
    If Not bIsActive Then
        If lvCategory.Visible Then lvCategory.SetFocus
        bIsActive = True
    End If
End Sub

```

```

Private Sub Form_Load()

    'Center window on screen
    Me.Left = (Screen.Width - Me.Width) / 2
    Me.Top = (Screen.Height - Me.Height) / 3

```

```

Me.Height = 3960
bExpanded = False
bIsActive = False

gbFmHelpLoaded = True

End Sub

```

```

Public Sub SetupData(Priority As Long)

    On Error GoTo ErrorHandler

    Dim TempConn As New ADODB.Connection
    Dim TempRS As New ADODB.Recordset
    Dim SQL As String
    Dim lLastPriority As Long
    Dim bAdded As Boolean
    Dim lNumCat As Long, lNumVuln As Long
    Dim Count As Long
    Dim lstItm As ListItem

    'Open connection
    TempConn.ConnectionString = Replace(gsDefaultConnString, "%PATH%", App.Path &
"\CCSVulnDB.mdb")
    TempConn.Open

    'Open recordset
    TempRS.CursorLocation = adUseServer
    TempRS.CursorType = adOpenStatic
    TempRS.LockType = adLockOptimistic

    SQL = "SELECT * FROM VulnData WHERE ID > 999 ORDER BY ID ASC"

    TempRS.Open SQL, TempConn, adCmdTable
    lLastPriority = -1
    lNumCat = 0
    Do While Not TempRS.EOF
        If (TempRS!ID Mod 1000 = 0) Then
            lNumCat = lNumCat + 1
            'If lNumCat = 19 Then lNumCat = 20
            lNumVuln = 0
            If lNumCat = 1 Then
                ReDim Cats(1 To lNumCat)
            Else
                ReDim Preserve Cats(1 To lNumCat)
            End If
            If lNumCat = 20 Then ReDim Cats(lNumCat - 1).Vulnerabilities(0)
            ReDim Cats(lNumCat).Vulnerabilities(0)

            Cats(lNumCat).Name = ReplaceAllCrLfFromText(" " & TempRS!VulDesS)
            Cats(lNumCat).ID = TempRS!ID

            If TempRS!Priority <> 999 Then
                lLastPriority = TempRS!Priority
            End If
        End If

        If (lLastPriority = TempRS!Priority) And (TempRS!ID <> TempRS!Priority) Then
            lNumVuln = lNumVuln + 1
            If lNumVuln = 1 Then
                ReDim Cats(lNumCat).Vulnerabilities(1 To lNumVuln)
            Else
                ReDim Preserve Cats(lNumCat).Vulnerabilities(1 To lNumVuln)
            End If

            Cats(lNumCat).Vulnerabilities(lNumVuln).Name =
ReplaceAllCrLfFromText(TempRS!VulDesS)
            Cats(lNumCat).Vulnerabilities(lNumVuln).ID = TempRS!ID

            Cats(lNumCat).Vulnerabilities(lNumVuln).RiskFactor = TempRS!RiskFactor
            Cats(lNumCat).Vulnerabilities(lNumVuln).Complexity = TempRS!Complexity
            Cats(lNumCat).Vulnerabilities(lNumVuln).Fixease = TempRS!Fixease
            Cats(lNumCat).Vulnerabilities(lNumVuln).Popularity = TempRS!Popularity
            Cats(lNumCat).Vulnerabilities(lNumVuln).RootCause = TempRS!RootCause
            Cats(lNumCat).Vulnerabilities(lNumVuln).Impact = TempRS!Impact
        End If
    Loop

```

```

        Cats(lNumCat).Vulnerabilities(lNumVuln).VulDesV = ReplaceAllCrLfFromText("" &
TempRS!VulDesV)
        Cats(lNumCat).Vulnerabilities(lNumVuln).Suggestion = ReplaceAllCrLfFromText("" &
TempRS!Suggestion)
        Cats(lNumCat).Vulnerabilities(lNumVuln).ManagerDesc = ReplaceAllCrLfFromText(""
& TempRS!ManagerDesc)

    End If

    TempRS.MoveNext
Loop

'Close
TempRS.Close
TempConn.Close

lvCategory.ListItems.Clear
For Count = 1 To lNumCat
    Set lstItm = lvCategory.ListItems.Add(, "Key" & CStr(Cats(Count).ID),
Cats(Count).ID)
    lstItm.SubItems(1) = Cats(Count).Name
    If lvCategory.ColumnHeaders("Name").Width < fmMain.TextWidth(Cats(Count).Name)
Then
        lvCategory.ColumnHeaders("Name").Width = fmMain.TextWidth(Cats(Count).Name) +
100
    End If

    If Priority = Cats(Count).ID Then
        lvCategory.ListItems(Count).Selected = True
        lvCategory.ListItems(Count).EnsureVisible
        Call lvCategory_ItemClick(lstItm)
    End If
Next Count

Quit:
Exit Sub

ErrorHandler:
Resume Quit
Resume

End Sub

Private Function ReplaceAllCrLfFromText(sFromText As String, Optional ReplaceWith As
String = "") As String

    Dim sText As String

    sText = sFromText
    If InStr(1, sText, vbCrLf) > 0 Then sText = Replace(sText, vbCrLf, ReplaceWith)
    If InStr(1, sText, vbLf) > 0 Then sText = Replace(sText, vbLf, ReplaceWith)
    If InStr(1, sText, vbCr) > 0 Then sText = Replace(sText, vbCr, ReplaceWith)
    ReplaceAllCrLfFromText = sText

End Function

Private Sub Form_Unload(Cancel As Integer)
    gbFmHelpLoaded = False
End Sub

Private Sub lvCategory_ItemClick(ByVal Item As MSComctlLib.ListItem)

    On Error GoTo ErrorHandler

    Dim lCat As Long, Count As Long
    Dim lstItm As ListItem

    If lvCategory.ListItems.Count < 1 Then Exit Sub
    For Count = 1 To lvCategory.ListItems.Count
        If lvCategory.ListItems(Count).Selected Then
            lCat = Count
            Exit For
        End If
    Next Count

    lvVulnerabilities.ListItems.Clear
    For Count = 1 To UBound(Cats(lCat).Vulnerabilities)

```

```

Set lstItm = lvVulnerabilities.ListItems.Add( "Key" &
CStr(Cats(lCat).Vulnerabilities(Count).ID), Cats(lCat).Vulnerabilities(Count).ID)
lstItm.SubItems(1) = Cats(lCat).Vulnerabilities(Count).Name
If lvVulnerabilities.ColumnHeaders("Name").Width <
fmMain.TextWidth(Cats(lCat).Vulnerabilities(Count).Name) Then
    lvVulnerabilities.ColumnHeaders("Name").Width =
fmMain.TextWidth(Cats(lCat).Vulnerabilities(Count).Name) + 100
End If
Next Count
If lvVulnerabilities.ListItems.Count > 0 Then
lvVulnerabilities.ListItems(1).Selected = True

Quit:
Exit Sub

ErrorHandler:
Resume Quit

End Sub

```

```

Private Sub lvVulnerabilities_ItemClick(ByVal Item As MSCComctlLib.ListItem)

Dim lTemp As Long
Dim lCat As Long, Count As Long
Dim sTemp As String

lCat = -1
sTemp = Replace(Item.Key, "Key", "")
lCat = Fix(CLng(sTemp) / 1000)

If lCat <= 0 Then Exit Sub

lTemp = Item.Index

If bExpanded Then
    lblID.Caption = Cats(lCat).Vulnerabilities(lTemp).ID
    lblDescription.Caption = Cats(lCat).Vulnerabilities(lTemp).Name

    Select Case Cats(lCat).Vulnerabilities(lTemp).RiskFactor
        Case 0: lblRiskFactor.Caption = "Low"
        Case 1: lblRiskFactor.Caption = "Medium"
        Case 2: lblRiskFactor.Caption = "High"
    End Select

    Select Case Cats(lCat).Vulnerabilities(lTemp).Complexity
        Case 0: lblComplexity.Caption = "N/A"
        Case 1: lblComplexity.Caption = "Low"
        Case 2: lblComplexity.Caption = "Medium"
        Case 3: lblComplexity.Caption = "High"
    End Select

    Select Case Cats(lCat).Vulnerabilities(lTemp).Fixease
        Case 0: lblFixease.Caption = "N/A"
        Case 1: lblFixease.Caption = "Trivial"
        Case 2: lblFixease.Caption = "Simple"
        Case 3: lblFixease.Caption = "Moderate"
        Case 4: lblFixease.Caption = "Difficult"
        Case 5: lblFixease.Caption = "Infeasable"
    End Select

    Select Case Cats(lCat).Vulnerabilities(lTemp).Popularity
        Case 0: lblPopularity.Caption = "N/A"
        Case 1: lblPopularity.Caption = "Obscure"
        Case 2: lblPopularity.Caption = "Widespread"
        Case 3: lblPopularity.Caption = "Popular"
    End Select

    Select Case Cats(lCat).Vulnerabilities(lTemp).RootCause
        Case 0: lblRootCause.Caption = "N/A"
        Case 1: lblRootCause.Caption = "Configuration"
        Case 2: lblRootCause.Caption = "Implementation"
        Case 3: lblRootCause.Caption = "Design"
    End Select

    txtImpact.Text = GetImpact(Cats(lCat).Vulnerabilities(lTemp).Impact)
    txtVDescription.Text = Cats(lCat).Vulnerabilities(lTemp).VulDesV
    txtSuggestion.Text = Cats(lCat).Vulnerabilities(lTemp).Suggestion

```



```

    txtManDescription.Text = Cats(lCat).Vulnerabilities(lTemp).ManagerDesc
End If

End Sub

```

```

Private Function GetImpact(lImpact As Long) As String

    Dim sTemp As String
    Dim lTemp As Long

    lTemp = lImpact
    If lTemp = 0 Then
        GetImpact = "N/A"
        Exit Function
    End If

    sTemp = ""
    Do While lTemp > 0
        If (lTemp >= vINTELLIGENCE) Then
            sTemp = sTemp & "Intelligence" & vbCrLf
            lTemp = lTemp - vINTELLIGENCE
        Else
            If (lTemp >= vAVAILABILITY) Then
                sTemp = sTemp & "Availability" & vbCrLf
                lTemp = lTemp - vAVAILABILITY
            Else
                If (lTemp >= vAUTHORIZATION) Then
                    sTemp = sTemp & "Authorization" & vbCrLf
                    lTemp = lTemp - vAUTHORIZATION
                Else
                    If (lTemp >= vDATA_INTEGRITY) Then
                        sTemp = sTemp & "Data Integrity" & vbCrLf
                        lTemp = lTemp - vDATA_INTEGRITY
                    Else
                        If (lTemp >= vACCOUNTABILITY) Then
                            sTemp = sTemp & "Accountability" & vbCrLf
                            lTemp = lTemp - vACCOUNTABILITY
                        Else
                            If (lTemp >= vCONFIDENTIALITY) Then
                                sTemp = sTemp & "Confidentiality" & vbCrLf
                                lTemp = lTemp - vCONFIDENTIALITY
                            Else
                                If (lTemp >= vSYSTEM_INTEGRITY) Then
                                    sTemp = sTemp & "System Integrity" & vbCrLf
                                    lTemp = lTemp - vSYSTEM_INTEGRITY
                                End If
                            End If
                        End If
                    End If
                End If
            End If
        End If
    Loop

    If Right$(sTemp, Len(vbCrLf)) = vbCrLf Then sTemp = Left$(sTemp, Len(sTemp) -
Len(vbCrLf))
    GetImpact = sTemp

End Function

```

B.1.4 The “frmMain” form

The design of this form is shown in figure B.5. This form is used as the start-up form from where all functions of the VF Prototype can be accessed..

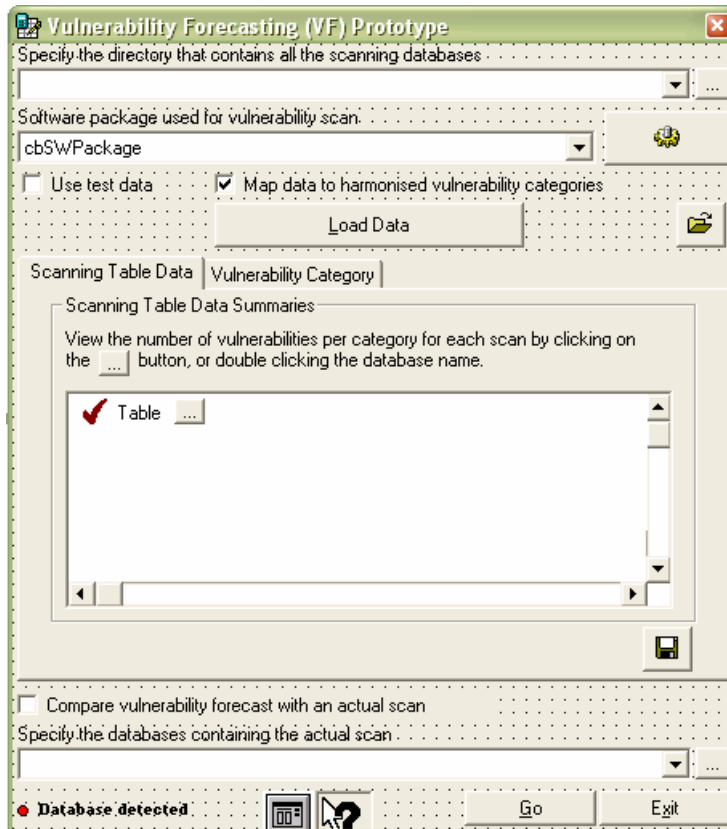


Figure B.5: The “frmMain” form

The source code for this form follows below.

```

Option Explicit
Dim nLastIndexSelected As Integer, nLastIndexSelectedCat As Integer
Dim lScrollingValue As Long, lScrollingValueCat As Long
Dim lButtonDown As Long
Dim bHelpClicked As Boolean

Private Sub cbDBDirs_Click()
    bHelpClicked = False
End Sub

Private Sub cbDBDirs_KeyUp(KeyCode As Integer, Shift As Integer)

    Dim sTemp As String

    If KeyCode = vbKeyReturn Then
        sTemp = cbDBDirs.Text
        If Right$(sTemp, 1) = "\" Then sTemp = Left$(sTemp, Len(sTemp) - 1)

        If sTemp = "" Then
            'Call MsgBox("Please specify a valid directory path!", vbOKOnly + vbInformation,
            "Database Path")
            Exit Sub
        End If

        If Dir(sTemp, vbDirectory) = "" Then

```

```

    Call MsgBox("Please specify a valid directory path!", vbOKOnly + vbInformation,
"Database Path")
    Exit Sub
End If

    cbDBDirs.AddItem sTemp
End If

End Sub

```

```

Private Function AddEntryToCbDBDirs(sAddEntry As String) As Boolean

```

```

    Dim sTemp As String
    Dim Count As Long
    Dim bOK As Boolean, bExists As Boolean

    sTemp = sAddEntry
    If Right$(sTemp, 1) = "\" Then sTemp = Left$(sTemp, Len(sTemp) - 1)

    If sTemp = "" Then
        Call MsgBox("Please specify a valid directory path!", vbOKOnly + vbInformation,
"Database Path")
        Exit Function
    End If

    If Dir(sTemp, vbDirectory) = "" Then
        Call MsgBox("Please specify a valid directory path!", vbOKOnly + vbInformation,
"Database Path")
        Exit Function
    End If

    'Check if entry already exists
    bOK = False
    bExists = False
    For Count = 1 To cbDBDirs.ListCount
        If LCase(cbDBDirs.List(Count - 1)) = LCase(sAddEntry) Then
            bExists = True
            bOK = True
            AddEntryToCbDBDirs = False
            Exit For
        End If
    Next Count

    If Not bOK Then
        cbDBDirs.AddItem sTemp
        cbDBDirs.ListIndex = cbDBDirs.NewIndex
        AddEntryToCbDBDirs = True
    Else
        AddEntryToCbDBDirs = bExists
    End If

End Function

```

```

Private Function AddEntryToCbActual(sAddEntry As String) As Boolean

```

```

    Dim sTemp As String
    Dim Count As Long
    Dim bOK As Boolean, bExists As Boolean

    sTemp = sAddEntry
    If Right$(sTemp, 1) = "\" Then sTemp = Left$(sTemp, Len(sTemp) - 1)

    'Check if entry already exists
    bOK = False
    bExists = False
    For Count = 1 To cbActual.ListCount
        If LCase(cbActual.List(Count - 1)) = LCase(sAddEntry) Then
            bExists = True
            bOK = True
            cbActual.ListIndex = (Count - 1)
            Exit For
        End If
    Next Count

    If Not bOK Then
        cbActual.AddItem sTemp
        cbActual.ListIndex = cbActual.NewIndex
    End If

```

```

    AddEntryToCbActual = True
Else
    AddEntryToCbActual = bExists
End If

End Function

Private Sub chkCompare_Click()
    Label4.Enabled = (chkCompare.Value = vbChecked)
    cbActual.Enabled = (chkCompare.Value = vbChecked)
    cmdSelectActual.Enabled = (chkCompare.Value = vbChecked)
    cmdGo.Enabled = (chkCompare.Value = vbChecked)
End Sub

Private Sub chkTest_Click()
    bHelpClicked = False
End Sub

Private Sub cmdDummy_Click()
    bHelpClicked = False
    If picDBsIn.Visible Then picDBsIn.SetFocus
End Sub

Private Sub cmdDummy_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
    Call DeselectAllTables
    Call DeselectAllCategories
End Sub

Private Sub cmdDummyCat_Click()
    bHelpClicked = False
    If picCategoriesIn.Visible Then picCategoriesIn.SetFocus
End Sub

Private Sub cmdDummyCat_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
    Call DeselectAllTables
    Call DeselectAllCategories
End Sub

Private Sub cmdExit_Click()
    Unload Me
End Sub

Private Sub cmdGo_Click()

    Dim Count As Long, Place As Long, lMaxGraphValue As Long, CountCats As Long
    Dim bContinue As Boolean
    Dim sDBNamePath As String, sDBName As String, sDBPath As String
    Dim PredictionsLB() As Long, PredictionsUB() As Long

    bContinue = True
    For Count = 1 To glNumVulnerabilityCategories
        'If Count <> 19 Then
            If Not Categories(Count).HasBeenSetup Then
                bContinue = False
                Exit For
            End If
        'End If
    Next Count

    If Not bContinue Then
        If MsgBox("Not all of the vulnerability scanning categories have been set up yet. A vulnerability category that has been set up successfully contains a red tick mark next to it. Do you want to continue anyway?", vbYesNoCancel + vbInformation, "View Results") <> vbYes Then Exit Sub
        bContinue = True
    End If

    sDBNamePath = cbActual.Text
    If sDBNamePath = "" Then
        bContinue = False
    Else
        Place = InStrRev(sDBNamePath, "\")
        If Place > 0 Then
            sDBName = Mid$(sDBNamePath, Place + 1)
            sDBPath = Left$(sDBNamePath, Place)
        End If
    End If

```

```

Else
    sDBName = sDBNamePath
    sDBPath = ""
End If

If Dir(sDBNamePath, vbArchive + vbHidden + vbNormal + vbReadOnly + vbSystem +
vbVolume) <> sDBName Then bContinue = False
End If

If Not bContinue Then
    Call MsgBox("Please specify a valid database file!", vbOKOnly + vbInformation,
"Comparing Data")
    If (cbActual.Visible) And (cbActual.Enabled) Then cbActual.SetFocus
    Exit Sub
End If

'Save directory path & file
If AddEntryToCbActual(sDBNamePath) Then
    Screen.MousePointer = vbHourglass

    If chkTest.Value = vbChecked Then
        'Test info
        ReDim ActualScan.VulnCount(1 To glNumVulnerabilityCategories)
        ReDim ActualScan.VulnID(1 To glNumVulnerabilityCategories)
        Call SetupDataForScan(16,
"40,0,0,0,0,0,0,7,0,0,0,0,0,0,0,23,1,44,0,0,67,0,3,16,19,144,0,2,0,1,0", True)
    Else
        'Read info from db
        ActualScan.TableName = sDBName
        Call GetNumberOfVulnerabilitiesForTable(sDBPath, ActualScan)
    End If

    ReDim PredictionsLB(1 To glNumVulnerabilityCategories)
    ReDim PredictionsUB(1 To glNumVulnerabilityCategories)
    For Count = 1 To glNumVulnerabilityCategories
        PredictionsLB(Count) = Categories(Count).Final_AmtVulns.Lowerbound
        PredictionsUB(Count) = Categories(Count).Final_AmtVulns.Upperbound
    Next Count

    'ActualScan.VulnCount & Predictions(Count)
    Call SavePathsInCbActualToText

    'Display Results
    Load fmGraphics

    With fmGraphics.GraphView1
        .Special_LineColor = Options.Prediction_LineColor
        .Prediction_LineColor = Options.Prediction_ColumnColor

        .Prediction_Heading_Top = "Forecast Bounds"
        .Prediction_Heading_Bottom = "Actual Scan"
        .Prediction_DisplayInfo = True

        .Heading = "Comparison Between Forecast and Actual Vulnerability Scan"
        .XAxis_Heading = "Vulnerability Category"
        .XAxis_Increment = 1
        .XAxis_Min = 0
        .XAxis_Max = glNumVulnerabilityCategories

        .YAxis_Heading = "Number of Vulnerabilities"
        .YAxis_Increment = 10
        .YAxis_Min = 0

        lMaxGraphValue = 100
        For CountCats = 1 To glNumDatabaseScans
            Do While ScanInfo(CountCats).MaxEntries > lMaxGraphValue
                lMaxGraphValue = lMaxGraphValue + 10
            Loop
            'If ScanInfo(Count).VulnCount(CountCats) > ScanInfo(Count).MaxEntries Then
            '    ScanInfo(Count).MaxEntries = ScanInfo(Count).VulnCount(CountCats)
            'End If
        Next CountCats
        For CountCats = 1 To glNumVulnerabilityCategories
            If ActualScan.MaxEntries < ActualScan.VulnCount(CountCats) Then
                ActualScan.MaxEntries = ActualScan.VulnCount(CountCats)
            Do While ActualScan.MaxEntries > lMaxGraphValue
                lMaxGraphValue = lMaxGraphValue + 10
            End If
        Next CountCats
    End With

```

```

        Loop
        Next CountCats
        .YAxis_Max = lMaxGraphValue
        .XAxis_Values = ""

        Call .DrawGraphWithPredictions(ActualScan.VulnCount, PredictionsLB,
PredictionsUB)
        End With

        fmGraphics.Show vbModal
    End If

    Screen.MousePointer = vbNormal

End Sub

Private Sub cmdHelp_Click()
    bHelpClicked = True
End Sub

Private Sub cmdHelp_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As
Single)
    Call DeselectAllTables
    Call DeselectAllCategories
End Sub

Private Sub cmdLoad_Click()

    Load fmSaveLoad
    fmSaveLoad.cmdSaveLoad.Caption = "&Load"
    fmSaveLoad.Show vbModal

End Sub

Public Sub cmdLoadData_Click()
    Call cmdLoadDataClick
End Sub

Public Sub cmdLoadDataClick(Optional bSetupCategoryInfo As Boolean = True)

    Dim sBuffer As String
    Dim Count As Long, CountCats As Long
    Dim sngTop As Single
    Dim nIndex As Integer

    If LCase(cbSWPackage.Text) = "<none>" Then
        Call MsgBox("You firsts have to set up the software information via the options!",
vbOKOnly + vbInformation)
        Exit Sub
    End If

    bHelpClicked = False

    'Get index
    nIndex = cbSWPackage.ItemData(cbSWPackage.ListIndex)
    SWPackageCurrent = SWPackageInfo(nIndex)

    'Set up vulnerability categories
    If chkMap.Value = vbChecked Then
        glNumVulnerabilityCategories = glCustomCategoriesNum
        lblInfo.Caption = "Loading vulnerability data and mapping to harmonised
vulnerability categories..."
    Else
        glNumVulnerabilityCategories = UBound(SWPackageCurrent.VulnDB.MainCategories)
        lblInfo.Caption = "Loading vulnerability info..."
    End If

    Call ClearForm(lblDBName.Count > 1)

    If (SWPackageCurrent.ScanningDB.TableName = "") Or
(SWPackageCurrent.ScanningDB.FieldName = "") Then
        Call MsgBox("You first have to set up the table and field name containing the
vulnerability scanning info via the options!", vbOKOnly + vbInformation)
        Exit Sub
    End If

    If (chkTest.Visible) And (chkTest.Value = vbChecked) Then

```

```

    Call LoadData
    Exit Sub
End If

tsMain.Tabs(1).Selected = True
tsMain.Visible = False
frmMain(0).Visible = False
frmMain(1).Visible = False

chkCompare.Visible = False
chkCompare.Value = vbUnchecked
Label4.Visible = False
cbActual.Visible = False
cmdSelectActual.Visible = False
cmdGo.Visible = False
cmdSave.Visible = False

pbLoading.Value = 0
frmLoading.Visible = True
Me.Refresh

sBuffer = cbDBDirs.Text
If (sBuffer = "") Then Exit Sub

If AddEntryToCbDBDirs(sBuffer) Then
    Screen.MousePointer = vbHourglass

    If (Right$(sBuffer, 1) <> "\") Then sBuffer = sBuffer & "\"
    glNumDatabaseScans = GetNumberOfDBsInDirectory(sBuffer, ScanInfo)

    If glNumDatabaseScans > 0 Then

        'Loop through databases (scans)
        sngTop = imgCheck(0).Top

        For Count = 1 To glNumDatabaseScans
            pbLoading.Value = Fix(Count / glNumDatabaseScans * 100)

            Call GetNumberOfVulnerabilitiesForTable(sBuffer, ScanInfo(Count))

            For CountCats = 1 To glNumVulnerabilityCategories
                If ScanInfo(Count).VulnCount(CountCats) > ScanInfo(Count).MaxEntries Then
                    ScanInfo(Count).MaxEntries = ScanInfo(Count).VulnCount(CountCats)
                End If
            Next CountCats

            'Add tables to picDBsIn
            Load imgCheck(Count)
            imgCheck(Count).Move imgCheck(0).Left, sngTop
            imgCheck(Count).Visible = True

            Load lblDBName(Count)
            lblDBName(Count).Caption = "" & ScanInfo(Count).TableName
            lblDBName(Count).Move lblDBName(0).Left, imgCheck(Count).Top +
            ((imgCheck(Count).Height - lblDBName(Count).Height) / 2)
            lblDBName(Count).Visible = True

            Load cmdSetup(Count)
            cmdSetup(Count).Move picDBsIn.Width - cmdSetup(Count).Width - 25,
            imgCheck(Count).Top, cmdSetup(0).Width, imgCheck(Count).Height
            cmdSetup(Count).Visible = True

            sngTop = imgCheck(Count).Top + imgCheck(Count).Height + 10
        Next Count
        pbLoading.Value = 100

        picDBsIn.Height = sngTop + 100
        picGap.Visible = False
        If (picDBsIn.Height > picDBs.Height) Then
            VScroll1.Visible = True
            picDBsIn.Width = VScroll1.Left
            If (picDBsIn.Width > picDBs.Width) Then
                HScroll1.Visible = True
                HScroll1.Width = picDBs.Width - VScroll1.Width
                VScroll1.Height = picDBs.Height - VScroll1.Height
                picGap.Move HScroll1.Left, VScroll1.Top
                picGap.Visible = True
            End If
        End If
    End If
End If

```



```

cmdSetup(Count).Move picDBsIn.Width - cmdSetup(Count).Width - 25,
imgCheck(Count).Top, cmdSetup(0).Width, imgCheck(Count).Height
cmdSetup(Count).Visible = True

sngTop = imgCheck(Count).Top + imgCheck(Count).Height + 10
Next Count

picDBsIn.Height = sngTop + 100
picGap.Visible = False
If (picDBsIn.Height > picDBs.Height) Then
    VScroll1.Visible = True
    picDBsIn.Width = VScroll11.Left
    If (picDBsIn.Width > picDBs.Width) Then
        HScroll11.Visible = True
        HScroll11.Width = picDBs.Width - VScroll11.Width
        VScroll1.Height = picDBs.Height - VScroll11.Height
        picGap.Move HScroll11.Left, VScroll11.Top
        picGap.Visible = True
    Else
        HScroll11.Visible = False
        VScroll1.Height = picDBs.Height - 60
    End If
Else
    picDBsIn.Width = VScroll11.Left + VScroll11.Width
    VScroll1.Visible = False
    If (picDBsIn.Width > picDBs.Width) Then
        HScroll11.Visible = True
        HScroll11.Width = picDBs.Width - 60
    Else
        HScroll11.Visible = False
    End If
End If

tsMain.Visible = True
frmMain(0).Visible = True
lScrollingValue = Abs(picDBsIn.Height - picDBs.Height) / 10
VScroll11.Max = Abs(picDBsIn.Height - picDBs.Height) / lScrollingValue

chkCompare.Visible = True
Label4.Enabled = False
Label4.Visible = True
cbActual.Enabled = False
cbActual.Visible = True
cmdSelectActual.Enabled = False
cmdSelectActual.Visible = True
cmdGo.Enabled = False
cmdGo.Visible = True

If bSetupCategoryInfo Then Call SetupCategoryInfo

End If

Screen.MousePointer = vbNormal

End Sub

```

```

Private Sub SetupDataForScan(ScanNumber As Long, NumbersString As String, Optional
bIsComparingDB As Boolean = False)

Dim Count As Long, Place As Long
Dim lNum As Long
Dim sTemp As String, sVal As String

If bIsComparingDB Then
    ActualScan.TableName = "Fuzzy" & Format(ScanNumber, "00") & ".mdb"
Else
    ScanInfo(ScanNumber).TableName = "Fuzzy" & Format(ScanNumber, "00") & ".mdb"
End If
lNum = 0
sTemp = NumbersString
Do While (lNum < glNumVulnerabilityCategories) And (sTemp <> "")
    lNum = lNum + 1
    Place = InStr(1, sTemp, ",")
    If Place > 0 Then
        sVal = Left$(sTemp, Place - 1)
        sTemp = Mid(sTemp, Place + 1)
    Else

```

```

        sVal = sTemp
        sTemp = ""
    End If
    If bIsComparingDB Then
        ActualScan.VulnCount(lNum) = CLng(sVal)
        If ActualScan.MaxEntries < ActualScan.VulnCount(lNum) Then ActualScan.MaxEntries
= ActualScan.VulnCount(lNum)
    Else
        ScanInfo(ScanNumber).VulnCount(lNum) = CLng(sVal)
        If ScanInfo(ScanNumber).MaxEntries < ScanInfo(ScanNumber).VulnCount(lNum) Then
ScanInfo(ScanNumber).MaxEntries = ScanInfo(ScanNumber).VulnCount(lNum)
    End If
    Loop
End Sub

Private Sub cmdLoadData_MouseMove(Button As Integer, Shift As Integer, X As Single, Y
As Single)
    Call DeselectAllTables
    Call DeselectAllCategories
End Sub

Private Sub cmdSave_Click()

    Load fmSaveLoad
    fmSaveLoad.cmdSaveLoad.Caption = "&Save"
    fmSaveLoad.Show vbModal

End Sub

Private Sub cmdSelectActual_Click()

    Dim sTemp As String
    Dim Place As Long

    sTemp = cbActual.Text
    If sTemp = "" Then
        sTemp = App.Path
    Else
        Place = InStrRev(sTemp, "\")
        sTemp = Left$(sTemp, Place - 1)
    End If

    cmdDlgActual.DefaultExt = sTemp
    cmdDlgActual.DialogTitle = "Select Database"
    cmdDlgActual.Filter = "Database Files|*.mdb"
    cmdDlgActual.InitDir = sTemp
    cmdDlgActual.ShowOpen

    sTemp = cmdDlgActual.FileName
    If sTemp <> "" Then
        cbActual.Text = sTemp
    End If

End Sub

Private Sub cmdSelectDBDir_Click()
    Dim lpIDList As Long
    Dim sBuffer As String
    Dim sDBFiles() As String

    bHelpClicked = False

    InitDir = ""
    If Right$(InitDir, 1) = "\" Then
        If Len(InitDir) > 3 Then
            InitDir = Left$(InitDir, Len(InitDir) - 1)
        End If
    End If

    gsFilter = ""
    gsStatusTextFound = "Database found"
    BrowseInfo.hWndOwner = Me.hwnd

    BrowseInfo.pIDLRoot = 0
    BrowseInfo.pszDisplayName = lpIDList

```



```

BrowseInfo.lpszTitle = lstrcat("Select database path", "")

BrowseInfo.ulFlags = BIF_STATUSTEXT

BrowseInfo.lpfncallback = GetProcAddress(AddressOf BrowseCallbackProc)
BrowseInfo.lParam = 0
lpIDList = SHBrowseForFolder(BrowseInfo)

If (lpIDList) Then
    sBuffer = Space(MAX_PATH)
    Call SHGetPathFromIDList(lpIDList, sBuffer)
    sBuffer = Left(sBuffer, InStr(sBuffer, vbNullChar) - 1)
    cbDBDirs.Text = sBuffer
End If

End Sub

Private Sub cmdSelectDBDir_MouseMove(Button As Integer, Shift As Integer, X As Single,
Y As Single)
    Call DeselectAllTables
    Call DeselectAllCategories
End Sub

Private Sub cmdSetup_Click(Index As Integer)

    Dim lMaxGraphValue As Long, Count As Long
    Dim sTemp As String

    Screen.MousePointer = vbHourglass

    bHelpClicked = False

    Load fmGraphics
    fmGraphics.GraphView1.Special_LineColor = Options.Prediction_LineColor
    fmGraphics.GraphView1.Prediction_LineColor = Options.Prediction_ColumnColor

    fmGraphics.GraphView1.Heading = "Vulnerabilities for Scan " & CStr(Index)
    fmGraphics.GraphView1.XAxis_Heading = "Vulnerability Category"
    fmGraphics.GraphView1.XAxis_Increment = 1
    fmGraphics.GraphView1.XAxis_Min = 0
    fmGraphics.GraphView1.XAxis_Max = glNumVulnerabilityCategories

    fmGraphics.GraphView1.YAxis_Heading = "Number of Vulnerabilities"
    fmGraphics.GraphView1.YAxis_Increment = 10
    fmGraphics.GraphView1.YAxis_Min = 0

    If ScanInfo(Index).MaxEntries > 100 Then
        lMaxGraphValue = 100
        Do While ScanInfo(Index).MaxEntries > lMaxGraphValue
            lMaxGraphValue = lMaxGraphValue + 10
        Loop
        fmGraphics.GraphView1.YAxis_Max = lMaxGraphValue
    Else
        fmGraphics.GraphView1.YAxis_Max = 100
    End If

    sTemp = ""
    For Count = 1 To UBound(SWPackageCurrent.VulnDB.MainCategories)
        sTemp = sTemp & SWPackageCurrent.VulnDB.MainCategories(Count).Number & ", "
    Next Count
    If Right$(sTemp, 1) = ", " Then sTemp = Left$(sTemp, Len(sTemp) - 1)
    fmGraphics.GraphView1.XAxis_Values = sTemp

    Call fmGraphics.GraphView1.DrawGraphColumns(ScanInfo(Index).VulnCount)

    If picDBsIn.Visible Then picDBsIn.SetFocus
    DoEvents
    fmGraphics.Show

End Sub

Private Sub cmdSetup_MouseMove(Index As Integer, Button As Integer, Shift As Integer,
X As Single, Y As Single)
    Call DeselectAllCategories
    Call DeselectAllTables
    lblDBName(Index).FontBold = True
End Sub

```

```

Private Sub cmdSetupCategory_Click(Index As Integer)

    Dim lMaxGraphValue As Long, Count As Long, CountIn As Long

    If bHelpClicked Then
        bHelpClicked = False
    End If

    Screen.MousePointer = vbHourglass

    gnCurCat = CLng(lblCategory(Index).Tag)

    Load fmSetup
    fmSetup.GraphView1.Special_LineColor = Options.Prediction_LineColor
    fmSetup.GraphView1.Prediction_LineColor = Options.Prediction_ColumnColor

    fmSetup.GraphView1.Heading = "Vulnerabilities for Vulnerability Category " &
    CStr(gnCurCat)
    fmSetup.GraphView1.XAxis_Heading = "Scan Number"
    fmSetup.GraphView1.XAxis_Increment = 1
    fmSetup.GraphView1.XAxis_Min = 0
    fmSetup.GraphView1.XAxis_Max = glNumDatabaseScans

    fmSetup.GraphView1.YAxis_Heading = "Number of Vulnerabilities"
    fmSetup.GraphView1.YAxis_Increment = 10
    fmSetup.GraphView1.YAxis_Min = 0

    lMaxGraphValue = 100
    If lMaxGraphValue < Categories(Index).MaxVulnerabilityValue Then lMaxGraphValue =
    Categories(Index).MaxVulnerabilityValue
    fmSetup.GraphView1.YAxis_Max = lMaxGraphValue

    Call fmSetup.GraphView1.DrawGraphColumns(Categories(Index).NumberOfVulnerabilities)

    Call fmSetup.SetupFormWithCategoryInfo(Index)

    If Categories(Index).DisplayResultOnGraph Then
        fmSetup.chkDisplay.Value = vbChecked
    Else
        fmSetup.chkDisplay.Value = vbUnchecked
    End If

    If picCategoriesIn.Visible Then picCategoriesIn.SetFocus
    fmSetup.Show

End Sub

Private Sub cmdSetupCategory_MouseMove(Index As Integer, Button As Integer, Shift As
Integer, X As Single, Y As Single)
    Call DeselectAllTables
    Call SelectCategory(Index)
End Sub

Private Sub cmdOptions_Click()
    If gbDBDetected Then
        Screen.MousePointer = vbHourglass
        fmOptions.Show vbModal
    End If
End Sub

Private Sub Form_Click()
    bHelpClicked = False
End Sub

Private Sub Form_DblClick()
    'If lButtonDown = (vbCtrlMask + vbShiftMask + vbAltMask) Then
    '    chkTest.Value = vbChecked
    '    chkTest.Visible = True
    'End If
End Sub

Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)
    lButtonDown = Shift
End Sub

Private Sub Form_Load()

```

```

'Center form on screen
Me.Left = (Screen.Width - Me.Width) / 2
Me.Top = (Screen.Height - Me.Height) / 2

'Initialize variables
glNumDatabaseScans = 0

If gbDEDetected Then
    shpConnected.FillColor = vbGreen
Else
    shpConnected.FillColor = vbRed
End If

cmdLoad.Visible = (Options.SaveLoad <> 0)

lScrollingValue = 150
VScroll11.Max = Abs(picDBsIn.Height - picDBs.Height) / lScrollingValue
Call LoadPreviousPathsIntocbDBDirs
Call SetupSWPackageCombo

End Sub

```

```

Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As
Single)
    Call DeselectAllTables
    Call DeselectAllCategories
End Sub

```

```

Private Sub Form_Unload(Cancel As Integer)
    Call UnloadApplication(False)
End Sub

```

```

Private Sub Frame1_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As
Single)
    Call DeselectAllTables
End Sub

```

```

Private Sub frmMain_Click(Index As Integer)
    bHelpClicked = False
End Sub

```

```

Private Sub frmMain_MouseMove(Index As Integer, Button As Integer, Shift As Integer, X
As Single, Y As Single)
    Call DeselectAllTables
    Call DeselectAllCategories
End Sub

```

```

Private Sub HScroll11_Change()
    'move picture up/down
    picDBsIn.Left = -(HScroll11.Value * lScrollingValue)
End Sub

```

```

Private Sub HScroll11_Scroll()
    'move picture up/down
    picDBsIn.Left = -(HScroll11.Value * lScrollingValue)
End Sub

```

```

Private Sub HScrollCat_Change()
    'move picture up/down
    picCategoriesIn.Left = -(HScrollCat.Value * lScrollingValueCat)
End Sub

```

```

Private Sub HScrollCat_Scroll()
    'move picture up/down
    picCategoriesIn.Left = -(HScrollCat.Value * lScrollingValueCat)
End Sub

```

```

Private Sub imgCheck_Click(Index As Integer)
    bHelpClicked = False
End Sub

```

```

Private Sub imgCheck_DblClick(Index As Integer)
    Call cmdSetup_Click(Index)
End Sub

```

```

Private Sub imgCheck_MouseMove(Index As Integer, Button As Integer, Shift As Integer,
X As Single, Y As Single)
    Call DeselectAllCategories
    Call SelectTable(Index)
End Sub

Private Sub imgCheckCategory_Click(Index As Integer)
    bHelpClicked = False
End Sub

Private Sub imgCheckCategory_DblClick(Index As Integer)
    Call cmdSetupCategory_Click(Index)
End Sub

Private Sub imgCheckCategory_MouseMove(Index As Integer, Button As Integer, Shift As
Integer, X As Single, Y As Single)
    Call DeselectAllTables
    Call SelectCategory(Index)
End Sub

Private Sub Label1_Click()
    bHelpClicked = False
End Sub

Private Sub Label2_Click()
    bHelpClicked = False
End Sub

Private Sub Label2_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As
Single)
    Call DeselectAllTables
    Call DeselectAllCategories
End Sub

Private Sub Label3_Click()
    bHelpClicked = False
End Sub

Private Sub Label3_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As
Single)
    Call DeselectAllTables
    Call DeselectAllCategories
End Sub

Private Sub lblCategory_Click(Index As Integer)

    Dim Count As Long

    If bHelpClicked Then
        Load fmHelp
        Call fmHelp.SetupData(CLng(Index) * 1000)
        fmHelp.Show vbModal
        Call SetAllCategoryLabelMousePointersToDefault
    End If

End Sub

Private Sub lblCategory_DblClick(Index As Integer)
    Call cmdSetupCategory_Click(Index)
End Sub

Private Sub lblCategory_MouseMove(Index As Integer, Button As Integer, Shift As
Integer, X As Single, Y As Single)

    Call DeselectAllTables
    If bHelpClicked Then
        Set lblCategory(Index).MouseIcon = picCursor.Picture
        lblCategory(Index).MousePointer = vbCustom
    Else
        Set lblCategory(Index).MouseIcon = Nothing
        lblCategory(Index).MousePointer = vbNormal
    End If
    Call SelectCategory(Index)

End Sub

Private Sub lblDBName_Click(Index As Integer)

```

```

    bHelpClicked = False
End Sub

Private Sub lblDBName_DblClick(Index As Integer)
    Call cmdSetup_Click(Index)
End Sub

Private Sub lblDBName_MouseMove(Index As Integer, Button As Integer, Shift As Integer,
X As Single, Y As Single)
    If gbFmSetupLoaded Then Exit Sub
    Call SelectTable(Index)
End Sub

Private Sub picCategories_Click()
    bHelpClicked = False
End Sub

Private Sub picCategories_MouseMove(Button As Integer, Shift As Integer, X As Single,
Y As Single)
    Call DeselectAllTables
    Call DeselectAllCategories
End Sub

Private Sub picCategoriesIn_Click()
    bHelpClicked = False
End Sub

Private Sub picCategoriesIn_MouseMove(Button As Integer, Shift As Integer, X As
Single, Y As Single)

    Dim Count As Integer
    Dim nIndex As Integer

    nIndex = -1
    For Count = 1 To imgCheckCategory.Count - 1
        'If Count <> 19 Then
            If (Y >= imgCheckCategory(Count).Top) And (Y <= (imgCheckCategory(Count).Top +
imgCheckCategory(Count).Height)) Then
                nIndex = Count
                Exit For
            End If
        'End If
    Next Count

    If nIndex < 0 Then
        Call DeselectAllCategories
    Else
        Call SelectCategory(nIndex)
    End If
End Sub

Private Sub picDBs_Click()
    bHelpClicked = False
End Sub

Private Sub picDBs_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As
Single)
    Call DeselectAllTables
    Call DeselectAllCategories
End Sub

Private Sub picDBsIn_Click()
    bHelpClicked = False
End Sub

Private Sub picDBsIn_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As
Single)

    Dim Count As Integer
    Dim nIndex As Integer

    nIndex = -1
    For Count = 1 To imgCheck.Count - 1
        If (Y >= imgCheck(Count).Top) And (Y <= (imgCheck(Count).Top +
imgCheck(Count).Height)) Then
            nIndex = Count
        End If
    Next Count
End Sub

```

```

        Exit For
    End If
Next Count

If nIndex < 0 Then
    Call DeselectAllTables
Else
    Call SelectTable(nIndex)
End If

End Sub

Private Sub picGap_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
    Call DeselectAllTables
End Sub

Private Sub LoadPreviousPathsIntocbDBDirs()

    Dim Count As Long
    Dim lIndex As Long

    lIndex = 0
    For Count = 1 To UBound(gsPreviousPaths)
        If InStr(1, gsPreviousPaths(Count), "-->") Then
            cbDBDirs.AddItem Trim(Replace(gsPreviousPaths(Count), "-->", ""))
            lIndex = Count
        Else
            cbDBDirs.AddItem gsPreviousPaths(Count)
        End If
    Next Count
    If cbDBDirs.ListCount > 0 Then cbDBDirs.ListIndex = 0

    For Count = 1 To UBound(gsPreviousComparePaths)
        cbActual.AddItem gsPreviousComparePaths(Count)
    Next Count
    If cbActual.ListCount > 0 Then cbActual.ListIndex = 0

End Sub

Private Sub SavePathsInCbDBDirsToText()

    Dim lNum As Long, Count As Long

    lNum = cbDBDirs.ListCount
    If lNum > 0 Then
        If lNum > 5 Then lNum = 5
        ReDim gsPreviousPaths(1 To lNum)
    Else
        ReDim gsPreviousPaths(0)
    End If
    Exit Sub
End If

    For Count = 1 To lNum
        If cbDBDirs.ListIndex = (Count - 1) Then
            gsPreviousPaths(Count) = cbDBDirs.List(Count - 1) & "-->"
        Else
            gsPreviousPaths(Count) = cbDBDirs.List(Count - 1)
        End If
    Next Count

End Sub

Private Sub SavePathsInCbActualToText()

    Dim lNum As Long, Count As Long

    lNum = cbActual.ListCount
    If lNum > 0 Then
        If lNum > 5 Then lNum = 5
        ReDim gsPreviousComparePaths(1 To lNum)
    Else
        ReDim gsPreviousComparePaths(0)
    End If
    Exit Sub
End If

    For Count = 1 To lNum

```

```

    gsPreviousComparePaths(Count) = cbActual.List(Count - 1)
Next Count

End Sub

Private Sub tsMain_Click()
    Call SetTab(tsMain, frmMain)
End Sub

Private Sub tsMain_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As
Single)
    Call DeselectAllTables
    Call DeselectAllCategories
End Sub

Private Sub VScroll1_Change()
    'move picture up/down
    picDBsIn.Top = -(VScroll1.Value * lScrollingValue)
End Sub

Private Sub VScroll1_Scroll()
    'move picture up/down
    picDBsIn.Top = -(VScroll1.Value * lScrollingValue)
End Sub

Private Sub VScrollCat_Change()
    'move picture up/down
    picCategoriesIn.Top = -(VScrollCat.Value * lScrollingValueCat)
End Sub

Private Sub VScrollCat_Scroll()
    'move picture up/down
    picCategoriesIn.Top = -(VScrollCat.Value * lScrollingValueCat)
End Sub

Private Sub DeselectAllTables()

    Dim Count As Long

    If gbFmSetupLoaded Then Exit Sub

    For Count = 1 To (lblDBName.Count - 1)
        lblDBName(Count).FontBold = False
    Next Count
    nLastIndexSelected = -1

End Sub

Private Sub DeselectAllCategories()

    On Error Resume Next

    Dim Count As Long

    If gbFmSetupLoaded Then Exit Sub

    'For Count = 1 To (lblCategory.Count - 1)
    For Count = 1 To (lblCategory.Count)
        lblCategory(Count).FontBold = False
    Next Count
    nLastIndexSelectedCat = -1

End Sub

Private Sub SetAllCategoryLabelMousePointersToDefault()

    Dim Count As Long

    For Count = 1 To (lblCategory.Count - 1)
        'If Count <> 19 Then
        Set lblCategory(Count).MouseIcon = Nothing
        lblCategory(Count).MousePointer = vbNormal
        'End If
    Next Count
    bHelpClicked = False

End Sub

```

```

Private Sub SelectTable(Index As Integer)
    If gbFmSetupLoaded Then Exit Sub
    If nLastIndexSelected = Index Then Exit Sub
    Call DeselectAllTables
    lblDBName(Index).FontBold = True
    nLastIndexSelected = Index
End Sub

Private Sub SelectCategory(Index As Integer)
    If gbFmSetupLoaded Then Exit Sub
    If nLastIndexSelectedCat = Index Then Exit Sub
    Call DeselectAllCategories
    lblCategory(Index).FontBold = True
    nLastIndexSelectedCat = Index
End Sub

Private Sub SetupCategoriesPicBox()

    Dim Count As Long
    Dim sngTop As Single

    sngTop = imgCheckCategory(0).Top
    For Count = 1 To glNumVulnerabilityCategories
        'If Count <> 19 Then
            'Add categories to picCategoriesIn
            Load imgCheckCategory(Count)
            imgCheckCategory(Count).Move imgCheckCategory(0).Left, sngTop
            imgCheckCategory(Count).Visible = False

            Load lblCategory(Count)
            lblCategory(Count).Caption = "Category " &
SWPackageCurrent.VulnDB.MainCategories(Count).Number 'CStr(Count)
            lblCategory(Count).Tag =
CStr(SWPackageCurrent.VulnDB.MainCategories(Count).Number)
            lblCategory(Count).Move lblCategory(0).Left, imgCheckCategory(Count).Top +
((imgCheckCategory(Count).Height - lblCategory(Count).Height) / 2)
            lblCategory(Count).Visible = True

            Load cmdSetupCategory(Count)
            cmdSetupCategory(Count).Move picCategoriesIn.Width - cmdSetupCategory(Count).Width
- 25, imgCheckCategory(Count).Top, cmdSetupCategory(0).Width,
imgCheckCategory(Count).Height
            cmdSetupCategory(Count).Visible = True

            sngTop = imgCheckCategory(Count).Top + imgCheckCategory(Count).Height + 10
        'End If
    Next Count

    If glNumVulnerabilityCategories > 0 Then
        picCategoriesIn.Height = sngTop + 100
        picGapCat.Visible = False
        If (picCategoriesIn.Height > picCategories.Height) Then
            VScrollCat.Visible = True
            picCategoriesIn.Width = VScrollCat.Left
            If (picCategoriesIn.Width > picCategories.Width) Then
                HScrollCat.Visible = True
                HScrollCat.Width = picCategories.Width - VScrollCat.Width
                VScrollCat.Height = picCategories.Height - VScrollCat.Height
                picGapCat.Move HScrollCat.Left, VScrollCat.Top
                picGapCat.Visible = True
            Else
                HScrollCat.Visible = False
                VScrollCat.Height = picCategories.Height - 60
            End If
        Else
            picCategoriesIn.Width = VScrollCat.Left + VScrollCat.Width
            VScrollCat.Visible = False
            If (picCategoriesIn.Width > picCategories.Width) Then
                HScrollCat.Visible = True
                HScrollCat.Width = picCategories.Width - 60
            Else
                HScrollCat.Visible = False
            End If
        End If

        lScrollingValueCat = Abs(picDBsIn.Height - picDBs.Height) / 10
    
```



```

    VScrollCat.Max = Abs(picCategoriesIn.Height - picCategories.Height) /
    lScrollingValueCat

    End If

End Sub

Private Sub SetupCategoryInfo()

    On Error GoTo ErrorHandler

    Dim Count As Long, CountScan As Long
    Dim lNumScans As Long, lNumAdjectives As Long

    If (glNumVulnerabilityCategories <= 0) Or (glNumDatabaseScans <= 0) Then Exit Sub

    ReDim Categories(1 To glNumVulnerabilityCategories)

    For Count = 1 To glNumVulnerabilityCategories

        Categories(Count).CategoryNumber = CLng(lblCategory(Count).Tag)
        ReDim Categories(Count).NumberOfVulnerabilities(1 To glNumDatabaseScans)

        For CountScan = 1 To glNumDatabaseScans
            Categories(Count).NumberOfVulnerabilities(CountScan) =
            ScanInfo(CountScan).VulnCount(Count)
            If Categories(Count).MaxVulnerabilityValue <
            Categories(Count).NumberOfVulnerabilities(CountScan) Then
                Categories(Count).MaxVulnerabilityValue =
                Categories(Count).NumberOfVulnerabilities(CountScan)
            End If
        Next CountScan

        Categories(Count).NumberOfGroups = 0
        ReDim Categories(Count).Groups(0)

        lNumAdjectives = UBound(gsAdjectives)
        If lNumAdjectives > 0 Then
            ReDim Categories(Count).Adjectives(1 To lNumAdjectives)
        Else
            ReDim Categories(Count).Adjectives(0)
        End If

        Categories(Count).HasBeenSetup = False
        Categories(Count).DisplayResultOnGraph = False
    Next Count

    For Count = 1 To glNumVulnerabilityCategories
        If Categories(Count).MaxVulnerabilityValue < 1 Then
            lblCategory(Count).ForeColor = vbRed
            imgCheckCategory(Count).Visible = True
            Categories(Count).HasBeenSetup = True
            Categories(Count).Final_AmtVulns.Lowerbound = 0
            Categories(Count).Final_AmtVulns.Upperbound = 0
        Else
            lblCategory(Count).ForeColor = vbButtonText
        End If
    Next Count

Quit:
    Exit Sub

ErrorHandler:
    Resume Quit

End Sub

Private Sub ClearForm(bClear As Boolean)

    Dim Count As Long

    If bClear Then
        For Count = (lblDBName.Count - 1) To 1 Step -1
            Unload imgCheck(Count)
            Unload lblDBName(Count)
            Unload cmdSetup(Count)
        Next Count
    End If

End Sub

```

```

End If

For Count = (lblCategory.Count - 1) To 1 Step -1
  'If Count <> 19 Then
    Unload imgCheckCategory(Count)
    Unload lblCategory(Count)
    Unload cmdSetupCategory(Count)
    'imgCheckCategory(Count).Visible = False
  'End If
Next Count

Call SetupCategoriesPictureBox

End Sub

```

```

Public Sub SetupSWPackageCombo()

  On Error Resume Next

  Dim lNum As Long, Count As Long

  lNum = 0
  lNum = UBound(SWPackageInfo)

  cbSWPackage.Clear
  If lNum > 0 Then
    For Count = 1 To lNum
      cbSWPackage.AddItem SWPackageInfo(Count).Name
      cbSWPackage.ItemData(cbSWPackage.NewIndex) = SWPackageInfo(Count).Number
    Next Count
  Else
    cbSWPackage.AddItem "<None>"
  End If
  If cbSWPackage.ListCount > 0 Then cbSWPackage.ListIndex = 0

End Sub

```

B.1.5 The “frmOptions” form

The design of this form is shown in figure B.6. This form is used to set up some options and aspects of the VF Prototype, for example, the adjective list and the harmonised vulnerability categories. There are also some options that can be set regarding the colour preference of the forecasting lines that indicate the vulnerability forecast range.

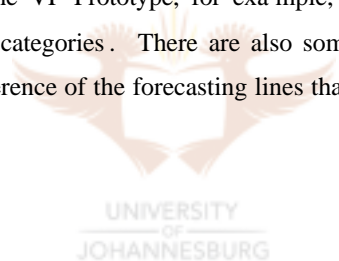


Figure B.6: The “frmOptions” form

The source code for this form follows below.

```

Option Explicit
Dim bChanges As Boolean

```

```

Private Sub cbVulnSWList_Click()

    Dim bEnable As Boolean
    Dim nTemp As Integer

    bEnable = ((cbVulnSWList.ListIndex >= 0) And (LCase(cbVulnSWList.Text) <> "<none>"))
    Label3.Enabled = bEnable
    Label4.Enabled = bEnable
    txtScanTableName.Enabled = bEnable
    txtScanFieldName.Enabled = bEnable
    cmdSelectDBDir.Enabled = bEnable

    If bEnable Then
        If Not gbBusyAddEdit Then
            nTemp = cbVulnSWList.ItemData(cbVulnSWList.ListIndex)
            SWPackageCurrent = SWPackageInfo(nTemp)
            txtScanTableName.Text = SWPackageCurrent.ScanningDB.TableName
            txtScanFieldName.Text = SWPackageCurrent.ScanningDB.FieldName
        End If
    Else

```

```

    txtScanTableName.Text = ""
    txtScanFieldName.Text = ""
End If

End Sub

Private Sub chkLoadSave_Click()
Options.SaveLoad = chkLoadSave.Value

If chkLoadSave.Value = vbChecked Then
If fmMain.tsMain.Visible Then fmMain.cmdSave.Visible = True
fmMain.cmdLoad.Visible = True
Else
If fmMain.tsMain.Visible Then fmMain.cmdSave.Visible = False
fmMain.cmdLoad.Visible = False
End If
End Sub

Private Sub cmdColors_Click(Index As Integer)

On Error GoTo ErrorHandler

cdColors.Color = picColors(Index).BackColor
cdColors.CancelError = True
cdColors.ShowColor

picColors(Index).BackColor = cdColors.Color
Select Case Index
Case 0: Options.Prediction_LineColor = cdColors.Color
Case 1: Options.Prediction_ColumnColor = cdColors.Color
End Select

Quit:
cmdOK.SetFocus
Exit Sub

ErrorHandler:
Resume Quit

End Sub

Private Sub cmdDefault_Click()
Options.Prediction_LineColor = vbRed
Options.Prediction_ColumnColor = &HFF8080 'light blue
picColors(0).BackColor = Options.Prediction_LineColor
picColors(1).BackColor = Options.Prediction_ColumnColor
cmdOK.SetFocus
End Sub

Private Sub cmdOK_Click()

Dim Count As Long, lNum As Long, lIndex As Long
Dim sTemp As String
Dim TempConn As New ADODB.Connection
Dim TempRS As New ADODB.Recordset

'Reload scanning software
With fmMain.cbSWPackage
If (cbVulnSWList.ListCount > 0) And (cbVulnSWList.Text <> "<None>") Then
sTemp = .Text
.Clear

lNum = cbVulnSWList.ListCount
If (lNum = 1) And (cbVulnSWList.Text = "<None>") Then
.AddItem "<None>"
.ItemData(.NewIndex) = -1
.ListIndex = 0
Else
lIndex = 0
For Count = 1 To lNum
.AddItem cbVulnSWList.List(Count - 1)
.ItemData(.NewIndex) = cbVulnSWList.ItemData(Count - 1)
If sTemp = cbVulnSWList.List(Count - 1) Then
lIndex = lNum - 1
End If
Next Count
.ListIndex = lIndex

```

```

End If

'Open connection
TempConn.ConnectionString = gsConnectionStringToMainDB
TempConn.Open

'Open recordset
Call OpenDBTable(TempConn, TempRS, "SELECT * FROM SWPackage WHERE Package_ID=" &
CStr(SWPackageCurrent.Number))
If Not TempRS.EOF Then
    TempRS!Scan_DB_TableName = "" & txtScanTableName.Text
    TempRS!Scan_DB_FieldName = "" & txtScanFieldName.Text
    TempRS!Scan_DB_Sample_DB_PathName = "" &
SWPackageCurrent.ScanningDB.SampleDB.DBPathName
    TempRS.Update
End If

'Close recordset and connection
TempRS.Close
TempConn.Close

End If
End With

Unload Me

End Sub

Private Sub cmdSelectDBDir_Click()

Dim nTemp As Integer
Dim Count As Long, lNum As Long

If (cbVulnSWList.ListIndex < 0) Or (LCase(cbVulnSWList.Text) = "<none>") Then Exit
Sub

nTemp = cbVulnSWList.ItemData(cbVulnSWList.ListIndex)
SWPackageCurrent = SWPackageInfo(nTemp)
Load fmSetupNames
fmSetupNames.Show vbModal
If ModalResult Then
    SWPackageInfo(nTemp) = SWPackageCurrent
End If

End Sub

Private Sub Form_Activate()
Screen.MousePointer = vbNormal
End Sub

Private Sub Form_Load()

'Center form on screen
Me.Left = (Screen.Width - Me.Width) / 2
Me.Top = (Screen.Height - Me.Height) / 2

gbBusyAddEdit = False
bChanges = False
Call SetupForm

gbFmOptionsLoaded = True

End Sub

Private Sub Form_Unload(Cancel As Integer)

Dim Count As Long, lNum As Long

If bChanges Then
lNum = lstAdjectives.ListCount
If lNum <= 0 Then
    ReDim gsAdjectives(0)
Else
    ReDim gsAdjectives(1 To lNum)

For Count = 1 To lNum
    gsAdjectives(Count) = lstAdjectives.List(Count - 1)

```

```

        Next Count
    End If

End If

    gbFmOptionsLoaded = False
End Sub

Private Sub lstAdjectives_DblClick()
    Call EditAdjective
End Sub

Private Sub lstAdjectives_KeyDown(KeyCode As Integer, Shift As Integer)
    Call DeleteAdjective
End Sub

Private Sub lvCategories_DblClick()
    Call EditCategory
End Sub

Private Sub lvCategories_KeyDown(KeyCode As Integer, Shift As Integer)
    Call DeleteCategory
End Sub

Private Sub tbAdjectives_ButtonClick(ByVal Button As MSComctlLib.Button)

    Select Case Button.Key
        Case "Add": Call AddAdjective
        Case "Delete": Call DeleteAdjective
        Case "Properties": Call EditAdjective
    End Select

End Sub

Private Sub tbCategories_ButtonClick(ByVal Button As MSComctlLib.Button)

    Select Case Button.Key
        Case "Add": Call AddCategory
        Case "Delete": Call DeleteCategory
        Case "Properties": Call EditCategory
        Case "Default": Call DefaultCategories
    End Select

End Sub

Private Sub tbSW_ButtonClick(ByVal Button As MSComctlLib.Button)

    Select Case Button.Key
        Case "Add": Call AddSWPackage
        Case "Delete": Call DeleteSWPackage
        Case "Properties": Call EditSWPackage
    End Select

End Sub

Private Sub SetupForm()

    On Error Resume Next

    Dim Count As Long, lNum As Long
    Dim li As ListItem
    Dim sngTempWidth As Single

    'Set up adjectives
    lstAdjectives.Clear
    For Count = 1 To UBound(gsAdjectives)
        lstAdjectives.AddItem gsAdjectives(Count)
        lstAdjectives.ItemData(lstAdjectives.NewIndex) = Count
    Next Count
    If lstAdjectives.ListCount > 0 Then lstAdjectives.ListIndex = 0

    'Set up colors
    picColors(0).BackColor = Options.Prediction_LineColor
    picColors(1).BackColor = Options.Prediction_ColumnColor
    chkLoadSave.Value = Options.SaveLoad

    'Set up custom categories

```

```

lvCategories.ListItems.Clear
sngTempWidth = 0
For Count = 1 To glCustomCategoriesNum
    Set li = lvCategories.ListItems.Add(, "Key" &
CStr(CustomCategoryInfo(Count).Number), CStr(CustomCategoryInfo(Count).Number))
    li.SubItems(1) = CustomCategoryInfo(Count).Description
    If sngTempWidth < fmMain.TextWidth(CustomCategoryInfo(Count).Description) Then
sngTempWidth = fmMain.TextWidth(CustomCategoryInfo(Count).Description)
Next Count
sngTempWidth = sngTempWidth + 500
If lvCategories.ColumnHeaders("CatDescr").Width < sngTempWidth Then
lvCategories.ColumnHeaders("CatDescr").Width = sngTempWidth
If lvCategories.ListItems.Count > 0 Then lvCategories.ListItems(1).Selected = True

'Set up software list
lNum = 0
lNum = UBound(SWPackageInfo)
cbVulnSWList.Clear
If lNum > 0 Then
    For Count = 1 To lNum
        cbVulnSWList.AddItem SWPackageInfo(Count).Name
        cbVulnSWList.ItemData(cbVulnSWList.NewIndex) = Count
    Next Count
Else
    cbVulnSWList.AddItem "<None>"
End If
If cbVulnSWList.ListCount > 0 Then cbVulnSWList.ListIndex = 0
End Sub

```

```

Private Sub AddAdjective()

Dim sAdjective As String
Dim Count As Long
Dim bAdd As Boolean

bChanges = True
sAdjective = InputBox("Enter new adjective", "Adding Adjective")
If sAdjective <> "" Then

    bAdd = True
    For Count = 0 To (lstAdjectives.ListCount - 1)
        If LCase(sAdjective) = LCase(lstAdjectives.List(Count)) Then
            Call MsgBox("An adjective with this name already exists!", vbOKOnly +
vbInformation, "Adding Adjective")
            bAdd = False
            Exit For
        End If
    Next Count

    If bAdd Then
        lstAdjectives.AddItem sAdjective
        lstAdjectives.ItemData(lstAdjectives.NewIndex) = lstAdjectives.ListCount
        lstAdjectives.ListIndex = lstAdjectives.NewIndex
    End If
End If
End Sub

```

```

Private Sub DeleteAdjective()

Dim sAdjective As String

If (lstAdjectives.ListIndex < 0) Or (lstAdjectives.ListCount < 1) Then Exit Sub

bChanges = True
sAdjective = lstAdjectives.List(lstAdjectives.ListIndex)
If MsgBox("Are you sure that you want to delete the adjective '" & sAdjective &
"?'", vbYesNoCancel + vbQuestion, "Deleting Adjective") = vbYes Then
    lstAdjectives.RemoveItem (lstAdjectives.ListIndex)
    If lstAdjectives.ListCount > 0 Then lstAdjectives.ListIndex = 0
End If
End Sub

```

```

Private Sub EditAdjective()

```

```

Dim sAdjective As String, sOld As String
Dim Count As Long
Dim bEdit As Boolean

If (lstAdjectives.ListIndex < 0) Or (lstAdjectives.ListCount < 1) Then Exit Sub

bChanges = True
sOld = lstAdjectives.List(lstAdjectives.ListIndex)
sAdjective = InputBox("Change adjective to:", "Editing Adjective", sOld)
If sAdjective <> "" Then

    bEdit = True
    For Count = 0 To (lstAdjectives.ListCount - 1)
        If (LCase(sAdjective) = LCase(lstAdjectives.List(Count))) And (sOld <>
lstAdjectives.List(Count)) Then
            Call MsgBox("An adjective with this name already exists!", vbOKOnly +
vbInformation, "Editing Adjective")
            bEdit = False
            Exit For
        End If
    Next Count

    If bEdit Then
        lstAdjectives.List(lstAdjectives.ListIndex) = sAdjective
    End If
End If

End Sub

```

```

Private Sub AddCategory()

Dim sCategory As String
Dim Count As Long
Dim bAdd As Boolean
Dim li As ListItem
Dim TempConn As New ADODB.Connection
Dim TempRS As New ADODB.Recordset

bChanges = True
sCategory = InputBox("Enter new category description", "Adding Category")
If sCategory <> "" Then

    bAdd = True
    For Count = 1 To lvCategories.ListItems.Count
        If LCase(sCategory) = LCase(lvCategories.ListItems(Count).SubItems(1)) Then
            Call MsgBox("A category with this description already exists!", vbOKOnly +
vbInformation, "Adding Category")
            bAdd = False
            Exit For
        End If
    Next Count

    Screen.MousePointer = vbHourglass

    If bAdd Then
        glCustomCategoriesNum = glCustomCategoriesNum + 1
        If glCustomCategoriesNum = 1 Then
            ReDim CustomCategoryInfo(1 To glCustomCategoriesNum)
        Else
            ReDim Preserve CustomCategoryInfo(1 To glCustomCategoriesNum)
        End If
        CustomCategoryInfo(glCustomCategoriesNum).Number = glCustomCategoriesNum
        CustomCategoryInfo(glCustomCategoriesNum).Description = sCategory

        Call AddCategorytoDB(CustomCategoryInfo(glCustomCategoriesNum).Number,
CustomCategoryInfo(glCustomCategoriesNum).Description)

        'Open connection
        TempConn.ConnectionString = gsConnectionStringToMainDB
        TempConn.Open

        'Open recordset

        Set li = lvCategories.ListItems.Add(, "Key" & (lvCategories.ListItems.Count +
1), CStr(lvCategories.ListItems.Count + 1))
        li.SubItems(1) = sCategory
    End If
End Sub

```



```

        li.Selected = True
        Call lvCategories.SelectedItem.EnsureVisible
    End If
End If

Screen.MousePointer = vbNormal

End Sub

```

```

Private Sub AddCategorytoDB(CatNumber As Long, CatName As String)

    On Error Resume Next

    Dim TempConn As New ADODB.Connection
    Dim TempRS As New ADODB.Recordset

    'Open connection
    TempConn.ConnectionString = gsConnectionStringToMainDB
    TempConn.Open

    'Open recordset
    Call OpenDBTable(TempConn, TempRS, "SELECT * FROM HVC WHERE HVC_Number=" &
    CatNumber)
    If TempRS.EOF Then
        TempRS.AddNew
    End If
    TempRS!HVC_Number = CatNumber
    TempRS!HVC_Name = "" & CatName
    TempRS.Update

End Sub

```

```

Private Sub DefaultCategories()

    Dim li As ListItem
    Dim Count As Long

    If MsgBox("Are you sure that you want to reload the default categories and delete
the current ones?", vbYesNoCancel + vbQuestion, "Default Categories") = vbYes Then
        Screen.MousePointer = vbHourglass

        lvCategories.ListItems.Clear
        Set li = lvCategories.ListItems.Add(, "Key1", 1)
        li.SubItems(1) = "Password Cracking and Sniffing"
        Set li = lvCategories.ListItems.Add(, "Key2", 2)
        li.SubItems(1) = "Network and System Information Gathering"
        Set li = lvCategories.ListItems.Add(, "Key3", 3)
        li.SubItems(1) = "User Enumeration and Information"
        Set li = lvCategories.ListItems.Add(, "Key4", 4)
        li.SubItems(1) = "Backdoors, Trojans and Remote Controlling"
        Set li = lvCategories.ListItems.Add(, "Key5", 5)
        li.SubItems(1) = "Gaining Unauthorised Access to Remote Connections & Services"
        Set li = lvCategories.ListItems.Add(, "Key6", 6)
        li.SubItems(1) = "Privilege and User Escalation"
        Set li = lvCategories.ListItems.Add(, "Key7", 7)
        li.SubItems(1) = "Spoofing or Masquerading"
        Set li = lvCategories.ListItems.Add(, "Key8", 8)
        li.SubItems(1) = "Miss-configurations"
        Set li = lvCategories.ListItems.Add(, "Key9", 9)
        li.SubItems(1) = "Denial-of-Service (DoS) and Buffer Overflows"
        Set li = lvCategories.ListItems.Add(, "Key10", 10)
        li.SubItems(1) = "Virusses and Worms"
        Set li = lvCategories.ListItems.Add(, "Key11", 11)
        li.SubItems(1) = "Hardware Specific"
        Set li = lvCategories.ListItems.Add(, "Key12", 12)
        li.SubItems(1) = "Software Specific and Updates"
        Set li = lvCategories.ListItems.Add(, "Key13", 13)
        li.SubItems(1) = "Security Policy Violations"
        'Set li = lvCategories.ListItems.Add(, "Key14", 14)
        'li.SubItems(1) = "Web Site or Organisational Defacement"
        'Set li = lvCategories.ListItems.Add(, "Key15", 15)
        'li.SubItems(1) = "Potential False Positives"
        lvCategories.ListItems(1).Selected = True
        Call lvCategories.SelectedItem.EnsureVisible

        glCustomCategoriesNum = 15
        ReDim CustomCategoryInfo(1 To glCustomCategoriesNum)
    End If
End Sub

```

```

    Call DeleteCategoryFromDB(-1)
    For Count = 1 To lvCategories.ListItems.Count
        Call AddCategorytoDB(CLng(lvCategories.ListItems.Count).Text),
lvCategories.ListItems.Count).SubItems(1)
        CustomCategoryInfo(Count).Number = CLng(lvCategories.ListItems(Count).Text)
        CustomCategoryInfo(Count).Description =
lvCategories.ListItems(Count).SubItems(1)
    Next Count

End If

Screen.MousePointer = vbNormal

End Sub

Private Sub DeleteCategory()

Dim sCategory As String
Dim lCatNum As Long

If (lvCategories.ListItems.Count < 1) Then Exit Sub
If (Not lvCategories.SelectedItem.Selected) Then Exit Sub

bChanges = True
sCategory = lvCategories.SelectedItem.SubItems(1)
If MsgBox("Are you sure that you want to delete the category '" & sCategory & "'",
vbYesNoCancel + vbQuestion, "Deleting Category") = vbYes Then
    Screen.MousePointer = vbHourglass

    lCatNum = CLng(lvCategories.SelectedItem.Text)
    Call DeleteCategoryFromDB(lCatNum)

    glCustomCategoriesNum = glCustomCategoriesNum - 1
    If glCustomCategoriesNum = 0 Then
        ReDim CustomCategoryInfo(0)
    Else
        ReDim Preserve CustomCategoryInfo(1 To glCustomCategoriesNum)
    End If

    Call lvCategories.ListItems.Remove(lvCategories.SelectedItem.Index)
    If (lvCategories.ListItems.Count > 0) Then lvCategories.ListItems(1).Selected =
True
    lvCategories.SelectedItem.EnsureVisible

End If
Screen.MousePointer = vbNormal

End Sub

Private Sub DeleteCategoryFromDB(CatNumber As Long)

On Error Resume Next

Dim TempConn As New ADODB.Connection
Dim TempRS As New ADODB.Recordset

'Open connection
TempConn.ConnectionString = gsConnectionStringToMainDB
TempConn.Open

If CatNumber < 0 Then
    TempConn.Execute "DELETE FROM HVC"
Else
    TempConn.Execute "DELETE FROM HVC WHERE HVC_Number=" & CatNumber
End If

'Close connectio
TempConn.Close

End Sub

Private Sub EditCategory()

Dim sCategory As String, sOld As String
Dim Count As Long
Dim bEdit As Boolean

```

```

Dim lIndex As Long

If (lvCategories.ListItems.Count < 1) Then Exit Sub
If (Not lvCategories.SelectedItem.Selected) Then Exit Sub

bChanges = True
sOld = lvCategories.SelectedItem.SubItems(1)
sCategory = InputBox("Change category description to:", "Editing Category", sOld)
If sCategory <> "" Then

    bEdit = True
    For Count = 1 To lvCategories.ListItems.Count
        If (LCase(sCategory) = LCase(lvCategories.ListItems(Count).SubItems(1))) And
(sOld <> lvCategories.ListItems(Count).SubItems(1)) Then
            Call MsgBox("A category with this description already exists!", vbOKOnly +
vbInformation, "Adding Category")
            bEdit = False
            Exit For
        End If
    Next Count

    If bEdit Then
        lIndex = CLng(Replace(lvCategories.SelectedItem.Key, "Key", ""))
        CustomCategoryInfo(lIndex).Description = sCategory
        lvCategories.SelectedItem.SubItems(1) = sCategory

        Call AddCategorytoDB(lIndex, sCategory)
    End If
End If

End Sub

```

```

Private Sub AddSWPackage()

    On Error GoTo ErrorHandler

    Dim lNumSWPackages As Long

    gbBusyAddEdit = True

    lNumSWPackages = UBound(SWPackageInfo)
    lNumSWPackages = lNumSWPackages + 1
    If lNumSWPackages = 1 Then
        ReDim SWPackageInfo(1 To lNumSWPackages)
    Else
        ReDim Preserve SWPackageInfo(1 To lNumSWPackages)
    End If
    Call ClearSWPackageInfo(SWPackageInfo(lNumSWPackages))
    SWPackageCurrent = SWPackageInfo(lNumSWPackages)

    Load fmSWSetup
    fmSWSetup.Show vbModal
    If ModalResult Then
        SWPackageInfo(lNumSWPackages) = SWPackageCurrent
    Else
        If lNumSWPackages = 1 Then
            ReDim SWPackageInfo(0)
        Else
            lNumSWPackages = lNumSWPackages - 1
            ReDim Preserve SWPackageInfo(1 To lNumSWPackages)
        End If
    End If

    gbBusyAddEdit = False
    Exit Sub

ErrorHandler:
    lNumSWPackages = 0
    Resume Next

End Sub

```

```

Private Sub DeleteSWPackage()

    Dim nTemp As Integer
    Dim Count As Long, lNum As Long

```

```

If (cbVulnSWList.ListIndex < 0) Or (LCase(cbVulnSWList.Text) = "<none>") Then Exit
Sub

If MsgBox("Are you sure that you want to delete the information of '" &
cbVulnSWList.Text & "'", vbYesNoCancel + vbQuestion) <> vbYes Then Exit Sub

nTemp = cbVulnSWList.ItemData(cbVulnSWList.ListIndex)

lNum = UBound(SWPackageInfo)
If lNum = 1 Then
    cbVulnSWList.Clear
    cbVulnSWList.AddItem "<None>"
    ReDim SWPackageInfo(0)
Else
    Call cbVulnSWList.RemoveItem(cbVulnSWList.ListIndex)

    For Count = nTemp To (lNum - 1)
        SWPackageInfo(Count) = SWPackageInfo(Count + 1)
    Next Count
    ReDim Preserve SWPackageInfo(1 To lNum - 1)
End If
If cbVulnSWList.ListCount > 0 Then cbVulnSWList.ListIndex = 0

End Sub

```

```

Private Sub EditSWPackage()

Dim nTemp As Integer
Dim Count As Long, lNum As Long

If (cbVulnSWList.ListIndex < 0) Or (LCase(cbVulnSWList.Text) = "<none>") Then Exit
Sub

gbBusyAddEdit = True
nTemp = cbVulnSWList.ItemData(cbVulnSWList.ListIndex)
SWPackageCurrent = SWPackageInfo(nTemp)
Load frmSWSetup
frmSWSetup.Show vbModal
If ModalResult Then
    SWPackageInfo(nTemp) = SWPackageCurrent
End If
gbBusyAddEdit = False

End Sub

```

B.1.6 The “frmSaveLoad” form

The design of this form is shown in figure B.7. This form is used to save and load profiles of settings as defined in the VF Prototype.



Figure B.7: The “frmSaveLoad” form

The source code for this form follows below.

```

Option Explicit

Private Type SaveLoadType
    ID As Long
    Name As String
    Date As Double
End Type
Dim SaveLoad() As SaveLoadType

Private Sub cmdCancel_Click()
    Unload Me
End Sub

Private Sub cmdDelete_Click()
    Call SaveData(True)
End Sub

Private Sub cmdSaveLoad_Click()

    On Error GoTo ErrorHandler

    Dim Count As Long, CountIn As Long, lExists As Long, lNum As Long, lNumCats As Long,
    lTempCounter As Long, lTempCounter1 As Long
    Dim FileNum As Long, WriteFileNum As Long, lLine As Long
    Dim sSplit() As String
    Dim InputData As String, sTemp As String
    Dim bOK As Boolean

    Screen.MousePointer = vbHourglass

    lLine = -1
    If cmdSaveLoad.Caption = "&Save" Then
        Call SaveData
    Else

        If txtName.Text = "" Then
            Screen.MousePointer = vbNormal
            Call MsgBox("You have to specify a name for the saved data!", vbOKOnly +
vbInformation, "Load Data")
            txtName.SetFocus
            Exit Sub
        End If

        bOK = False
        For Count = 1 To lvNames.ListItems.Count
            If LCase(lvNames.ListItems(Count).Text) = LCase(txtName.Text) Then
                bOK = True
                Exit For
            End If
        Next Count
        If Not bOK Then
            Screen.MousePointer = vbNormal
            Call MsgBox("The saved data's name that you specified does not exist!", vbOKOnly
+ vbInformation, "Load Data")
            txtName.SetFocus
            Exit Sub
        End If

        Me.Hide

        'Open file
        FileNum = FreeFile
        Open (App.Path & "\SavedData.vpl") For Input As #FileNum

        If Not EOF(FileNum) Then
            'Read Next line
            Line Input #FileNum, InputData

            lNum = 0
            If InputData = "SAVED_NAMES" Then
                Do
                    'Read Next line

```

```

Line Input #FileNum, InputData

If InputData <> "END OF SAVED_NAMES" Then
    sSplit = Split(InputData, "|||")
    If LCase(sSplit(1)) = LCase(txtName.Text) Then
        lNum = CLng(sSplit(0))
    End If
End If
Loop While (InputData <> "END OF SAVED_NAMES") And (lNum = 0)
End If

If (lNum <= 0) Or (EOF(FileNum)) Then Screen.MousePointer = vbNormal: Exit Sub

Do While Not EOF(FileNum)
    'Read Next line
    Line Input #FileNum, InputData

    If InputData = "DATA FOR ID " & CStr(lNum) Then
        Do
            'Read Next line
            Line Input #FileNum, InputData

            If InStr(1, InputData, "|||") Then
                sSplit = Split(InputData, "|||")

                Select Case UCase(sSplit(0))
                    Case "DB PATH": fmMain.cbDBDirs.Text = sSplit(1)
                    Case "SW PACKAGE": fmMain.cbSWPackage.Text = sSplit(1)
                    Case "MAP": fmMain.chkMap.Value = CInt(sSplit(1))

                    Case "CATEGORIES"
                        lNumCats = CLng(sSplit(1))
                        ReDim Categories(1 To lNumCats)
                        lTempCounter = 0

                        Do
                            'Read Next line
                            Line Input #FileNum, InputData

                            If InputData <> "END CATEGORIES" Then
                                sSplit = Split(InputData, "|||")

                                Select Case sSplit(0)
                                    Case "CatNum"
                                        lTempCounter = lTempCounter + 1
                                        Categories(lTempCounter).CategoryNumber = CLng(sSplit(1))
                                    Case "NumOfVulns"
                                        sTemp = sSplit(1)
                                        sSplit = Split(sTemp, ",")
                                        lNumCats = (UBound(sSplit) - LBound(sSplit)) + 1
                                        ReDim Categories(lTempCounter).NumberOfVulnerabilities(1
To lNumCats)

                                        For Count = 1 To lNumCats
                                            If sSplit(Count - 1) = "" Then
                                                Categories(lTempCounter).NumberOfVulnerabilities(Count) = 0
                                            Else
                                                Categories(lTempCounter).NumberOfVulnerabilities(Count) = CLng(sSplit(Count - 1))
                                            End If
                                        Next Count

                                    Case "MaxVulnVal":
                                        Categories(lTempCounter).MaxVulnerabilityValue = CLng(sSplit(1))
                                    Case "NumOfGroups"
                                        Categories(lTempCounter).NumberOfGroups = CLng(sSplit(1))
                                        If Categories(lTempCounter).NumberOfGroups > 0 Then
                                            ReDim Categories(lTempCounter).Groups(1 To
Categories(lTempCounter).NumberOfGroups)
                                        End If

                                        CountIn = 0
                                        Do
                                            'Read Next line
                                            Line Input #FileNum, InputData
                                            If InputData <> "END OF GROUPS" Then

```

```

sSplit = Split(InputData, "|||")
If sSplit(0) = "Group" Then
    CountIn = CountIn + 1
    sSplit = Split(sSplit(1), ",")
    If UBound(sSplit) <> 12 Then Screen.MousePointer =
vbNormal: Exit Sub

Categories(lTempCounter).Groups(CountIn).ScanFrom =
CLng(sSplit(0))
Categories(lTempCounter).Groups(CountIn).ScanTo =
CLng(sSplit(1))
Categories(lTempCounter).Groups(CountIn).Adjective =
sSplit(2)

Categories(lTempCounter).Groups(CountIn).VulnerabilityFrom = CLng(sSplit(3))
Categories(lTempCounter).Groups(CountIn).VulnerabilityTo = CLng(sSplit(4))
Categories(lTempCounter).Groups(CountIn).VulnerabilityTranslatedFrom = CDbL(sSplit(5))
Categories(lTempCounter).Groups(CountIn).VulnerabilityTranslatedTo = CDbL(sSplit(6))
Categories(lTempCounter).Groups(CountIn).Cx_From =
CDbl(sSplit(7))
Categories(lTempCounter).Groups(CountIn).Cx_To =
CDbl(sSplit(8))

Categories(lTempCounter).Groups(CountIn).Mu_Lowerbound = CDbL(sSplit(9))
Categories(lTempCounter).Groups(CountIn).Mu_Upperbound = CDbL(sSplit(10))
Categories(lTempCounter).Groups(CountIn).MIN_Cx_Mu_LB = CDbL(sSplit(11))
Categories(lTempCounter).Groups(CountIn).MIN_Cx_Mu_UB = CDbL(sSplit(12))
End If
End If
Loop Until InputData = "END OF GROUPS"

Case "Adjectives"
    If CLng(sSplit(1)) < 1 Then Screen.MousePointer =
vbNormal: Exit Sub
    ReDim Categories(lTempCounter).Adjectives(1 To
CLng(sSplit(1)))

    CountIn = 0
    Do
        'Read Next line
        Line Input #FileNum, InputData
        If InputData <> "END OF ADJECTIVES" Then
            sSplit = Split(InputData, "|||")
            If Left$(sSplit(0), 9) = "Adjective" Then
                CountIn = CountIn + 1
                sSplit = Split(sSplit(1), ",")
                If UBound(sSplit) <> 3 Then Screen.MousePointer =
vbNormal: Exit Sub

UNIVERSITY
OF

Categories(lTempCounter).Adjectives(CountIn).LowerOperator = CLng(sSplit(0))
Categories(lTempCounter).Adjectives(CountIn).LowerValue = sSplit(1)
Categories(lTempCounter).Adjectives(CountIn).UpperOperator = CLng(sSplit(2))
Categories(lTempCounter).Adjectives(CountIn).UpperValue = sSplit(3)
End If
End If
Loop Until InputData = "END OF ADJECTIVES"

Case "Rule_Name": Categories(lTempCounter).Rule_Name =
sSplit(1)
Case "Rule_Value": Categories(lTempCounter).Rule_Value =
sSplit(1)

Case "HasBeenSetup": Categories(lTempCounter).HasBeenSetup =
CBool(CLng(sSplit(1)))
Case "Membership_Op1":
Categories(lTempCounter).Membership_Op1 = CLng(sSplit(1))

```

```

                Case "Membership_Devider":
Categories(lTempCounter).Membership_Devider = CLng(sSplit(1))
                Case "Membership_Op2":
Categories(lTempCounter).Membership_Op2 = CLng(sSplit(1))
                Case "Membership_Op3":
Categories(lTempCounter).Membership_Op3 = CLng(sSplit(1))
                Case "Membership_To": Categories(lTempCounter).Membership_To
= CLng(sSplit(1))
                Case "Membership_Op4":
Categories(lTempCounter).Membership_Op4 = CLng(sSplit(1))

                Case "MAXofMIN_Cx_Mu"
sSplit = Split(sSplit(1), ",")
Categories(lTempCounter).MAXofMIN_Cx_Mu.Lowerbound =
Cdbl(sSplit(0))
Categories(lTempCounter).MAXofMIN_Cx_Mu.Upperbound =
Cdbl(sSplit(1))

                Case "Final"
sSplit = Split(sSplit(1), ",")
Categories(lTempCounter).Final.Lowerbound =
Cdbl(sSplit(0))
Categories(lTempCounter).Final.Upperbound =
Cdbl(sSplit(1))

                Case "Final_AmtVulns"
sSplit = Split(sSplit(1), ",")
Categories(lTempCounter).Final_AmtVulns.Lowerbound =
CLng(sSplit(0))
Categories(lTempCounter).Final_AmtVulns.Upperbound =
CLng(sSplit(1))

                Case "DisplayResultOnGraph":
Categories(lTempCounter).DisplayResultOnGraph = CBool(CLng(sSplit(1)))
                End Select
                End If
                Loop While InputData <> "END CATEGORIES"

                End Select
                Else
                If InputData <> "END DATA" Then
                Select Case InputData
                End Select
                End If
                End If

                Loop Until InputData = "END DATA"
                End If
                Loop
                End If

                Call fmMain.cmdLoadDataClick(False)

                For Count = 1 To (fmMain.lblCategory.Count - 1)
                fmMain.imgCheckCategory(Count).Visible = (Categories(Count).HasBeenSetup)

                If Categories(Count).MaxVulnerabilityValue < 1 Then
                fmMain.lblCategory(Count).ForeColor = vbRed
                Else
                fmMain.lblCategory(Count).ForeColor = vbButtonText
                End If
                Next Count
                End If
                Close #FileNum
                Screen.MousePointer = vbNormal
                Unload Me

                Exit Sub

ErrorHandler:
                If lLine = 0 Then lTempCounter = 0
                If lLine = 2 Then lTempCounter1 = 0
                Resume Next

                End Sub

Private Sub Form_Load()

```



```

Me.Left = (Screen.Width - Me.Width) / 2
Me.Top = (Screen.Height - Me.Height) / 2

Call ReadSavedNames

End Sub

```

```

Private Sub SaveData(Optional bSaveWithoutCurrent As Boolean = False)

    On Error GoTo ErrorHandler

    Dim Count As Long, CountIn As Long, lExists As Long, lNum As Long, lTempCounter As
Long, lTempCounter1 As Long
    Dim FileNum As Long, WriteFileNum As Long, lLine As Long
    Dim sSplit() As String
    Dim InputData As String, sTemp As String
    Dim OldNew() As PredictionType

    lLine = -1
    If txtName.Text = "" Then
        Screen.MousePointer = vbNormal
        If bSaveWithoutCurrent Then
            Call MsgBox("You have to specify the name of the saved data to be deleted!",
vbOKOnly + vbInformation, "Delete Data")
        Else
            Call MsgBox("You have to specify a name for the saved data!", vbOKOnly +
vbInformation, "Save Data")
        End If
        txtName.SetFocus
        Exit Sub
    End If

    If Not bSaveWithoutCurrent Then
        If InStr(1, txtName.Text, "\") Or InStr(1, txtName.Text, "/") Or InStr(1,
txtName.Text, ":") Or InStr(1, txtName.Text, "**") Or InStr(1, txtName.Text, "?") Or
InStr(1, txtName.Text, Chr(34)) Or InStr(1, txtName.Text, "<") Or InStr(1,
txtName.Text, ">") Or InStr(1, txtName.Text, "|") Then
            Screen.MousePointer = vbNormal
            Call MsgBox("The name for the saved data cannot contain any of the following
characters:" & vbCrLf & "\ / : * ? "" "" < > |", vbOKOnly + vbInformation, "Save Data")
            txtName.SetFocus
            Exit Sub
        End If
    End If

    lExists = -1
    For Count = 1 To lvNames.ListItems.Count
        If lvNames.ListItems(Count).Text = txtName.Text Then
            lExists = Count
            Exit For
        End If
    Next Count
    If lExists > 0 Then
        Screen.MousePointer = vbNormal
        If bSaveWithoutCurrent Then
            Call MsgBox("Are you sure that you want to delete the selected data?",
vbYesNoCancel + vbQuestion, "Delete Data") <> vbYes Then Exit Sub
        Else
            Call MsgBox("Are you sure that you want to replace the selected data with the new
data?", vbYesNoCancel + vbQuestion, "Save Data") <> vbYes Then Exit Sub
        End If
        Screen.MousePointer = vbHourglass
        lExists = CLng(Replace(lvNames.ListItems(lExists).Key, "Key", ""))
    Else
        If bSaveWithoutCurrent Then
            Screen.MousePointer = vbNormal
            Call MsgBox("You have to specify the name of the saved data to be deleted!",
vbOKOnly + vbInformation, "Delete Data")
            txtName.SetFocus
            Exit Sub
        End If
    End If

    'Open Write file
    WriteFileNum = FreeFile
    Open (App.Path & "\Temp.vpl") For Output Access Write As #WriteFileNum

```

```

Print #WriteFileNum, "SAVED_NAMES"

lNum = 0
For Count = 1 To lvNames.ListItems.Count
  If lvNames.ListItems(Count).Key <> ("Key" & lExists) Then
    lNum = lNum + 1
    If lNum = 1 Then
      ReDim OldNew(1 To lNum)
    Else
      ReDim Preserve OldNew(1 To lNum)
    End If
    OldNew(lNum).Lowerbound = CLng(Replace(lvNames.ListItems(Count).Key, "Key", ""))
    OldNew(lNum).Upperbound = lNum
    Print #WriteFileNum, lNum & "|||" & lvNames.ListItems(Count).Text & "|||" &
Cdbl(CDate(lvNames.ListItems(Count).SubItems(1)))
  End If
Next Count
If Not bSaveWithoutCurrent Then
  lNum = lNum + 1
  Print #WriteFileNum, CStr(lNum) & "|||" & txtName.Text & "|||" & Cdbl(Now)
End If

Print #WriteFileNum, "END OF SAVED_NAMES"

'Open file
FileNum = FreeFile
Open (App.Path & "\SavedData.vpl") For Input As #FileNum

If Not EOF(FileNum) Then
  Do
    'Read Next line
    Line Input #FileNum, InputData
    Loop Until InputData = "END OF SAVED_NAMES"

  If Not EOF(FileNum) Then
    Do
      'Read Next line
      Line Input #FileNum, InputData

      If (Left$(InputData, 12) <> "DATA FOR ID ") Then
        Do Until (Left$(InputData, 12) = "DATA FOR ID ") Or (EOF(FileNum))
          'Read Next line
          Line Input #FileNum, InputData
        Loop
      End If
      If InputData = "DATA FOR ID " & CStr(lExists) Then
        Do
          'Read Next line
          Line Input #FileNum, InputData
          Loop While InputData <> "END DATA"
        Else
          If (Left$(InputData, 12) = "DATA FOR ID ") Then
            For Count = 1 To UBound(OldNew)
              If OldNew(Count).Lowerbound = CLng(Mid(InputData, 13)) Then
                Print #WriteFileNum, "DATA FOR ID " & CStr(OldNew(Count).Upperbound)
              End If
            Next Count
          Else
            Print #WriteFileNum, InputData
          End If
        Do
          'Read Next line
          Line Input #FileNum, InputData
          Print #WriteFileNum, InputData
          Loop While InputData <> "END DATA"
        End If
      Loop Until EOF(FileNum)
    End If
  End If

  If Not bSaveWithoutCurrent Then
    Print #WriteFileNum, "DATA FOR ID " & CStr(lNum)
    Print #WriteFileNum, "DB PATH|||" & fmMain.cbDBDirs.Text
    Print #WriteFileNum, "SW PACKAGE|||" & fmMain.cbSWPackage.Text
    Print #WriteFileNum, "MAP|||" & fmMain.chkMap.Value
  
```

```

lLine = 0
lTempCounter = UBound(Categories)
lLine = 1

Print #WriteFileNum, "CATEGORIES|||" & lTempCounter
For Count = 1 To lTempCounter
  With Categories(Count)
    Print #WriteFileNum, "CatNum|||" & .CategoryNumber

    lLine = 2
    lTempCounter1 = UBound(.NumberOfVulnerabilities)
    lLine = 3
    sTemp = ""
    For CountIn = 1 To lTempCounter1
      sTemp = sTemp & .NumberOfVulnerabilities(CountIn) & ","
    Next CountIn
    If Right$(sTemp, 1) = "," Then sTemp = Left$(sTemp, Len(sTemp) - 1)
    Print #WriteFileNum, "NumOfVulns|||" & sTemp

    Print #WriteFileNum, "MaxVulnVal|||" & .MaxVulnerabilityValue
    Print #WriteFileNum, "NumOfGroups|||" & .NumberOfGroups

    lLine = 2
    lTempCounter1 = UBound(.Groups)
    lLine = 3
    For CountIn = 1 To lTempCounter1
      sTemp = .Groups(CountIn).ScanFrom & ","
      sTemp = sTemp & .Groups(CountIn).ScanTo & ","
      sTemp = sTemp & .Groups(CountIn).Adjective & ","
      sTemp = sTemp & .Groups(CountIn).VulnerabilityFrom & ","
      sTemp = sTemp & .Groups(CountIn).VulnerabilityTo & ","
      sTemp = sTemp & .Groups(CountIn).VulnerabilityTranslatedFrom & ","
      sTemp = sTemp & .Groups(CountIn).VulnerabilityTranslatedTo & ","
      sTemp = sTemp & .Groups(CountIn).Cx_From & ","
      sTemp = sTemp & .Groups(CountIn).Cx_To & ","
      sTemp = sTemp & .Groups(CountIn).Mu_Lowerbound & ","
      sTemp = sTemp & .Groups(CountIn).Mu_Upperbound & ","
      sTemp = sTemp & .Groups(CountIn).MIN_Cx_Mu_LB & ","
      sTemp = sTemp & .Groups(CountIn).MIN_Cx_Mu_UB
      Print #WriteFileNum, "Group|||" & sTemp
    Next CountIn
    Print #WriteFileNum, "END OF GROUPS"

    lLine = 2
    lTempCounter1 = UBound(.Adjectives)
    lLine = 3
    Print #WriteFileNum, "Adjectives|||" & lTempCounter1
    For CountIn = 1 To lTempCounter1
      sTemp = .Adjectives(CountIn).LowerOperator & ","
      sTemp = sTemp & .Adjectives(CountIn).LowerValue & ","
      sTemp = sTemp & .Adjectives(CountIn).UpperOperator & ","
      sTemp = sTemp & .Adjectives(CountIn).UpperValue
      Print #WriteFileNum, "Adjective" & CountIn & "|||" & sTemp
    Next CountIn
    Print #WriteFileNum, "END OF ADJECTIVES"

    Print #WriteFileNum, "Rule_Name|||" & .Rule_Name
    Print #WriteFileNum, "Rule_Value|||" & .Rule_Value

    Print #WriteFileNum, "HasBeenSetup|||" & CLng(.HasBeenSetup)
    Print #WriteFileNum, "Membership_Op1|||" & .Membership_Op1
    Print #WriteFileNum, "Membership_Devider|||" & .Membership_Devider
    Print #WriteFileNum, "Membership_Op2|||" & .Membership_Op2
    Print #WriteFileNum, "Membership_Op3|||" & .Membership_Op3
    Print #WriteFileNum, "Membership_To|||" & .Membership_To
    Print #WriteFileNum, "Membership_Op4|||" & .Membership_Op4

    Print #WriteFileNum, "MAXofMIN_Cx_Mu|||" & .MAXofMIN_Cx_Mu.Lowerbound & "," &
    .MAXofMIN_Cx_Mu.Upperbound
    Print #WriteFileNum, "Final|||" & .Final.Lowerbound & "," & .Final.Upperbound
    Print #WriteFileNum, "Final_AmtVulns|||" & .Final_AmtVulns.Lowerbound & "," &
    .Final_AmtVulns.Upperbound

    Print #WriteFileNum, "DisplayResultOnGraph|||" & CLng(.DisplayResultOnGraph)
  End With
Next Count
Print #WriteFileNum, "END CATEGORIES"

```

```

    Print #WriteFileNum, "END DATA"
End If

Close #WriteFileNum
Close #FileNum

Call Kill((App.Path & "\SavedData.vpl"))
Name (App.Path & "\Temp.vpl") As (App.Path & "\SavedData.vpl")

txtName.Text = ""
Call ReadSavedNames
Screen.MousePointer = vbNormal

Exit Sub

ErrorHandler:
If lLine = 0 Then lTempCounter = 0
If lLine = 2 Then lTempCounter1 = 0
Resume Next

End Sub

Private Sub ReadSavedNames()

Dim FileNum As Long, Count As Long
Dim InputData As String, sLeftHS As String, sRightHS As String
Dim lNum As Long, lNumCount As Long
Dim sSplit() As String
Dim bOK As Boolean
Dim li As ListItem

'Open file
FileNum = FreeFile

'Create text file if does not exist
If Dir(App.Path & "\SavedData.vpl", vbArchive + vbHidden + vbNormal + vbReadOnly +
vbSystem + vbVolume) <> "SavedData.vpl" Then
'Create file
Open (App.Path & "\SavedData.vpl") For Output Access Write As #FileNum

Print #FileNum, "SAVED_NAMES"
Print #FileNum, "END OF SAVED_NAMES"

'Close file
Close #FileNum
End If

'Open file for reading
Open (App.Path & "\SavedData.vpl") For Input As #FileNum

lNum = 0
bOK = EOF(FileNum)
Do While Not bOK
'Read Next line
Line Input #FileNum, InputData

If InputData = "SAVED_NAMES" Then
Do
'Read Next line
Line Input #FileNum, InputData

If InputData <> "END OF SAVED_NAMES" Then
lNum = lNum + 1
If lNum = 1 Then
ReDim SaveLoad(1 To lNum)
Else
ReDim Preserve SaveLoad(1 To lNum)
End If

sSplit = Split(InputData, "|||")
SaveLoad(lNum).ID = CLng(sSplit(0))
SaveLoad(lNum).Name = sSplit(1)
SaveLoad(lNum).Date = CDb1(sSplit(2))
Else
bOK = True
End If

```

```

        Loop Until InputData = "END OF SAVED_NAMES"
    End If
    Loop
    'Close file
    Close #FileNum
    lvNames.ListItems.Clear
    For Count = 1 To lNum
        Set li = lvNames.ListItems.Add(, "Key" & CStr(SaveLoad(Count).ID),
    SaveLoad(Count).Name)
        li.SubItems(1) = CDate(SaveLoad(Count).Date)
    Next Count

End Sub

Private Sub lvNames_Click()

    Dim Count As Long
    Dim bOK As Boolean
    If lvNames.ListItems.Count < 1 Then Exit Sub
    bOK = False
    For Count = 1 To lvNames.ListItems.Count
        If lvNames.ListItems(Count).Selected Then
            bOK = True
            Exit For
        End If
    Next Count
    If Not bOK Then Exit Sub
    txtName.Text = lvNames.SelectedItem.Text

End Sub

Private Sub lvNames_DblClick()
    Call cmdSaveLoad_Click
End Sub

Private Sub txtName_KeyDown(KeyCode As Integer, Shift As Integer)
    If KeyCode = vbKeyReturn Then
        Call cmdSaveLoad_Click
    End If
End Sub

```

B.1.7 The “frmSelectCats” form

The design of this form is shown in figure B.8. This form is used as part of setting up the harmonised vulnerability categories.

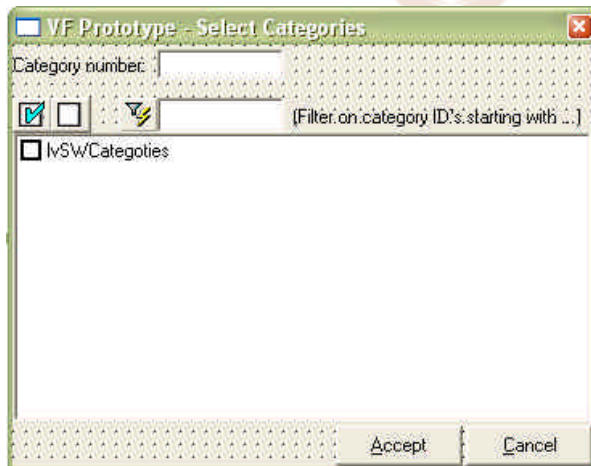


Figure B.8: The “frmSelectCats” form

The source code for this form follows below.

```

Option Explicit
Dim FilterArray() As CustomCategoryInfoType

```

```

Private Sub cmdAccept_Click()

    Dim Count As Long, lTemp As Long
    Dim sTemp As String

    sTemp = txtNumber.Text
    If Not IsNumeric(sTemp) Then
        Call MsgBox("You have to specify a valid numeric number for the category!",
vbOKOnly + vbInformation, "Incorrect Number")
        txtNumber.SetFocus
    Exit Sub
    End If

    For Count = 1 To fmSWSetup.lvMainCats.ListItems.Count
        If Not fmSWSetup.lvMainCats.ListItems(Count).Selected Then
            If sTemp = fmSWSetup.lvMainCats.ListItems(Count).Text Then
                Call MsgBox("This number is already assigned to another category!", vbOKOnly +
vbInformation, "Incorrect Number")
                txtNumber.SetFocus
            Exit Sub
            End If
        End If
    Next Count

    sTemp = ""
    For Count = 1 To lvSWCategories.ListItems.Count
        If lvSWCategories.ListItems(Count).Checked Then
            sTemp = sTemp & lvSWCategories.ListItems(Count).Text & ","
        End If
    Next Count

    If sTemp <> "" Then
        sTemp = Left$(sTemp, Len(sTemp) - 1)
        Call OptimizeCommaSeperatedNumbers(sTemp)
    End If

    fmSWSetup.lvMainCats.SelectedItem.Text = txtNumber.Text
    fmSWSetup.lvMainCats.SelectedItem.SubItems(3) = sTemp

    Call fmSWSetup.ReorderMainCategories
    Me.Hide
    cmdFilter.Value = vbUnchecked
    txtFilter.Text = ""
    Call RestoreOriginalCategories

End Sub

```

```

Private Sub cmdCancel_Click()
    Me.Hide
    cmdFilter.Value = vbUnchecked
    txtFilter.Text = ""
    Call RestoreOriginalCategories
End Sub

```

```

Private Sub cmdCheck_Click()
    Call CheckAll
End Sub

```

```

Private Sub cmdFilter_Click()

    Dim lNumEntries As Long, Count As Long, CountIn As Long
    Dim sTemp As String
    Dim li As ListItem
    Dim TempFilter() As CustomCategoryInfoType

    Screen.MousePointer = vbHourglass

    If cmdFilter.Value = vbChecked Then
        sTemp = txtFilter.Text
    End If

```

```

If Not IsNumeric(sTemp) Then
    Screen.MousePointer = vbNormal
    Call MsgBox("Not a valid filter!", vbOKOnly + vbInformation, "Filter")
    Exit Sub
End If

Count = 0
Do
    Count = Count + 1
    If Count <= lvSWCategories.ListItems.Count Then
        If Left$(lvSWCategories.ListItems(Count).Text, Len(sTemp)) <> sTemp Then
            Call lvSWCategories.ListItems.Remove(Count)
            Count = Count - 1
        End If
    End If
Loop While (Count <= lvSWCategories.ListItems.Count)

txtFilter.Enabled = False

Else

    Call RestoreOriginalCategories

    txtFilter.Enabled = True
End If

If lvSWCategories.Visible Then lvSWCategories.SetFocus

Screen.MousePointer = vbNormal

End Sub

```

```

Private Sub cmdUncheck_Click()
    Call CheckAll(False)
End Sub

```

```

Private Sub Form_Activate()

    Dim lNumEntries As Long, Count As Long

    If Me.Visible Then
        lNumEntries = lvSWCategories.ListItems.Count
        If lNumEntries < 1 Then Exit Sub

        ReDim FilterArray(1 To lNumEntries)
        For Count = 1 To lNumEntries
            FilterArray(Count).ID = CLng(lvSWCategories.ListItems(Count).Text)
            FilterArray(Count).Description = lvSWCategories.ListItems(Count).SubItems(1)
            FilterArray(Count).Key = lvSWCategories.ListItems(Count).Key

            If lvSWCategories.ListItems(Count).Checked Then
                FilterArray(Count).Number = 1
            Else
                FilterArray(Count).Number = 0
            End If
        Next Count

        txtNumber.SelStart = 0
        txtNumber.SelLength = Len(txtNumber.Text)
        txtNumber.SetFocus
    End If
End Sub

```

```

Private Sub Form_Load()

    Me.Left = (Screen.Width - Me.Width) / 2
    Me.Top = (Screen.Height - Me.Height) / 2

End Sub

```

```

Private Sub CheckAll(Optional bCheck As Boolean = True)

    Dim Count As Long

    For Count = 1 To lvSWCategories.ListItems.Count
        If bCheck Then
            If lvSWCategories.ListItems(Count).Selected Then

```

```

        lvSWCategories.ListItems(Count).Checked = bCheck
    End If
Else
    lvSWCategories.ListItems(Count).Checked = bCheck
End If
Next Count
lvSWCategories.SetFocus
End Sub

```

```

Private Sub RestoreOriginalCategories()

    Dim lNumEntries As Long, Count As Long, CountIn As Long
    Dim sTemp As String
    Dim li As ListItem
    Dim TempFilter() As CustomCategoryInfoType

    ReDim TempFilter(0)
    If lvSWCategories.ListItems.Count > 0 Then
        ReDim TempFilter(1 To lvSWCategories.ListItems.Count)
        For Count = 1 To lvSWCategories.ListItems.Count
            TempFilter(Count).ID = CLng(lvSWCategories.ListItems(Count).Text)
            TempFilter(Count).Description = lvSWCategories.ListItems(Count).SubItems(1)
            TempFilter(Count).Key = lvSWCategories.ListItems(Count).Key

            If lvSWCategories.ListItems(Count).Checked Then
                TempFilter(Count).Number = 1
            Else
                TempFilter(Count).Number = 0
            End If
        Next Count
    End If

    Call lvSWCategories.ListItems.Clear

    lNumEntries = UBound(FilterArray)
    For Count = 1 To lNumEntries
        Set li = lvSWCategories.ListItems.Add(, FilterArray(Count).Key,
        FilterArray(Count).ID)
        li.SubItems(1) = FilterArray(Count).Description

        If (FilterArray(Count).Number = 1) Then li.Checked = True

        For CountIn = 1 To UBound(TempFilter)
            If TempFilter(CountIn).ID = FilterArray(Count).ID Then
                If (TempFilter(CountIn).Number = 1) Then li.Checked = True
            End If
        Next CountIn
    Next Count
    If lvSWCategories.ListItems.Count > 0 Then
        lvSWCategories.ListItems(1).Selected = True
        Call lvSWCategories.ListItems(1).EnsureVisible
    End If
End Sub

```

```

Private Sub txtFilter_KeyDown(KeyCode As Integer, Shift As Integer)
    If KeyCode = vbKeyReturn Then
        cmdFilter.Value = vbChecked
        Call cmdFilter_Click
    End If
End Sub

```

B.1.8 The “frmSetup” form

The design of this form is shown in figure B.9. This form is used to set up the mapping table, fuzzy groups, and membership function in the process of doing a vulnerability forecast.

VF Prototype - Vulnerability Forecasting

General Setup

Mapping Table

Adjective	Lower Bound	Upper Bound
Much more than	x cl	x cl

Fuzzy Groups

Number of groups: 5 (Navigate with ↑ and ↓ keys, or ▲ and ▼ buttons - make sure a drop down list does not have focus)

Group 1: In [] to [] scan(s) [cbAdjectives] to [] vulnerabilities were found

Group 2: In [] to [] scan(s) [cbAdjectives] to [] vulnerabilities were found

Group 3: In [] to [] scan(s) [cbAdjectives] to [] vulnerabilities were found

Group 4: In [] to [] scan(s) [cbAdjectives] to [] vulnerabilities were found

Group 5: In [] to [] scan(s) [cbAdjectives] to [] vulnerabilities were found

Membership Function

$$C(x) = \begin{cases} 0 & \text{if } x = 0 \\ \frac{x}{c} & \text{if } 0 < x < c \\ 1 & \text{if } x = 100 \end{cases}$$

Rule:

IF the amount of vulnerabilities found for vulnerability category n is [cbRule] [0]

THEN increase expertise for this specific vulnerability category.

<< Back Next >> Cancel

Figure B.9: The "frmSetup" form

The source code for this form follows below.

```

Option Explicit
Dim btnDown As Long
Dim bCbOperatorsGotFocus As Boolean
Dim nTabIndex As Integer

Private Sub cbAdjectives_GotFocus(Index As Integer)
    bCbOperatorsGotFocus = True
End Sub

Private Sub cbAdjectives_KeyDown(Index As Integer, KeyCode As Integer, Shift As Integer)
    If (KeyCode = vbKeyDelete) Or (KeyCode = vbKeyEscape) Then
        cbAdjectives(Index).ListIndex = -1
    End If
End Sub

Private Sub cbAdjectives_LostFocus(Index As Integer)
    bCbOperatorsGotFocus = False
End Sub

Private Sub cbOperatorLower_Click(Index As Integer)

    If cbOperatorLower(Index).ListIndex < 0 Then Exit Sub

    If cbOperatorLower(Index).ItemData(cbOperatorLower(Index).ListIndex) = xINFINITY
Then
        lblXLower(Index).Visible = False
    
```

```

    lblValLower(Index).Visible = False
Else
    If Not lblXLower(Index).Visible Then lblXLower(Index).Visible = True
    If Not lblValLower(Index).Visible Then lblValLower(Index).Visible = True
End If

End Sub

Private Sub cbOperatorLower_GotFocus(Index As Integer)
    bCbOperatorsGotFocus = True
End Sub

Private Sub cbOperatorLower_KeyDown(Index As Integer, KeyCode As Integer, Shift As Integer)
    If (KeyCode = vbKeyDelete) Or (KeyCode = vbKeyEscape) Then
        cbOperatorLower(Index).ListIndex = -1
    End If
End Sub

Private Sub cbOperatorLower_LostFocus(Index As Integer)
    bCbOperatorsGotFocus = False
    DoEvents
End Sub

Private Sub cbOperatorUpper_Click(Index As Integer)

    If cbOperatorUpper(Index).ItemData(cbOperatorUpper(Index).ListIndex) = xINFINITY
Then
        lblXUpper(Index).Visible = False
        lblValUpper(Index).Visible = False
    Else
        If Not lblXUpper(Index).Visible Then lblXUpper(Index).Visible = True
        If Not lblValUpper(Index).Visible Then lblValUpper(Index).Visible = True
    End If

End Sub

Private Sub cbOperatorUpper_KeyDown(Index As Integer, KeyCode As Integer, Shift As Integer)
    If (KeyCode = vbKeyDelete) Or (KeyCode = vbKeyEscape) Then
        cbOperatorUpper(Index).ListIndex = -1
    End If
End Sub

Private Sub cbRule_GotFocus()
    bCbOperatorsGotFocus = True
End Sub

Private Sub cbRule_KeyDown(KeyCode As Integer, Shift As Integer)
    If (KeyCode = vbKeyDelete) Or (KeyCode = vbKeyEscape) Then
        cbRule.ListIndex = -1
    End If
End Sub

Private Sub cbRule_LostFocus()
    bCbOperatorsGotFocus = False
End Sub

Private Sub chkDisplay_Click()
    GraphView1.Special_Lowerbound = Categories(gnCurCat).Final_AmtVulns.Lowerbound
    GraphView1.Special_Upperbound = Categories(gnCurCat).Final_AmtVulns.Upperbound
    GraphView1.Special_Display = (chkDisplay.Value = vbChecked)

    Categories(gnCurCat).DisplayResultOnGraph = (chkDisplay.Value = vbChecked)
End Sub

Private Sub cmdAccept_Click()
    Call cmdAcceptClick
End Sub

Private Sub cmdAcceptClick(Optional bShowFrame As Boolean = True)

    On Error Resume Next

    Dim Count As Long, lNum As Long

    If CheckInfo Then

```

```

Select Case nTabIndex
Case 1
'Groups
Categories(gnCurCat).NumberOfGroups = CLng(lblNumGroups.Caption)

ReDim Categories(gnCurCat).Groups(1 To Categories(gnCurCat).NumberOfGroups)
For Count = 1 To Categories(gnCurCat).NumberOfGroups
Categories(gnCurCat).Groups(Count).ScanFrom = CLng(txtScanFrom(Count -
1).Text)
Categories(gnCurCat).Groups(Count).ScanTo = CLng(txtScanTo(Count - 1).Text)
Categories(gnCurCat).Groups(Count).Adjective = cbAdjectives(Count - 1).Text
Categories(gnCurCat).Groups(Count).VulnerabilityFrom =
CLng(txtVulnFrom(Count - 1).Text)
Categories(gnCurCat).Groups(Count).VulnerabilityTo = CLng(txtVulnTo(Count -
1).Text)
Next Count

'Adjectives
lNum = UBound(gsAdjectives)
ReDim Categories(gnCurCat).Adjectives(1 To lNum)
For Count = 1 To lNum
If cbOperatorLower(Count).ListIndex >= 0 Then
Categories(gnCurCat).Adjectives(Count).LowerOperator =
cbOperatorLower(Count).ItemData(cbOperatorLower(Count).ListIndex)
Else
Categories(gnCurCat).Adjectives(Count).LowerOperator = -1
End If
Categories(gnCurCat).Adjectives(Count).LowerValue = lblValLower(Count).Text

If cbOperatorUpper(Count).ListIndex >= 0 Then
Categories(gnCurCat).Adjectives(Count).UpperOperator =
cbOperatorUpper(Count).ItemData(cbOperatorUpper(Count).ListIndex)
Else
Categories(gnCurCat).Adjectives(Count).UpperOperator = -1
End If
Categories(gnCurCat).Adjectives(Count).UpperValue = lblValUpper(Count).Text
Next Count

'Rule
Categories(gnCurCat).Rule_Name = cbRule.Text
Categories(gnCurCat).Rule_Value = txtRule.Text

Call DoTranslationTab

cmdAccept.Caption = "&Next >>"
If TabStrip1.Tabs.Count < 2 Then
TabStrip1.Tabs.Add 2, "Translation", "Translation"
End If
nTabIndex = 2
If bShowFrame Then
TabStrip1.Tabs(2).Selected = True
Call SetTab(TabStrip1, frmWizard)
frmGraph.ZOrder 0
End If
cmdBack.Visible = True

Case 2
cmdAccept.Caption = "Finish"
If TabStrip1.Tabs.Count < 3 Then
TabStrip1.Tabs.Add 3, "Conclusion", "Conclusion"
TabStrip1.Tabs.Add 3, "Forecast", "Forecast"
End If
nTabIndex = 3
TabStrip1.Tabs(3).Selected = True
Call SetTab(TabStrip1, frmWizard)
frmGraph.ZOrder 0
cmdBack.Visible = True

Case 3
Categories(gnCurCat).HasBeenSetup = True

If Not fmMain.imgCheckCategory(gnCurCat).Visible Then
fmMain.imgCheckCategory(gnCurCat).Visible = True
End If

gnCurCat = -1

```

```

        Call UnloadThisForm
    End Select

End If

End Sub

Private Sub cmdBack_Click()
    Call cmdBackClick
End Sub

Private Sub cmdBackClick(Optional bShowFrame As Boolean = True)
    If nTabIndex > 1 Then
        nTabIndex = nTabIndex - 1
        If bShowFrame Then
            TabStrip1.Tabs(nTabIndex).Selected = True
            Call SetTab(TabStrip1, frmWizard)
        End If
        If nTabIndex = 1 Then cmdBack.Visible = False
        cmdAccept.Caption = "&Next >>"
    Else
        cmdBack.Visible = False
    End If
End Sub

Private Sub cmdCancel_Click()
    Call UnloadThisForm
End Sub

Private Sub cmdDown_Click()

    Dim lNum As Long

    lNum = CLng(lblNumGroups.Caption)
    lNum = lNum - 1
    If lNum < 1 Then lNum = 1
    lblNumGroups.Caption = CStr(lNum)
    picFocus.SetFocus
    Call EnableDisableGroups

End Sub

Private Sub cmdUp_Click()

    Dim lNum As Long

    lNum = CLng(lblNumGroups.Caption)
    lNum = lNum + 1
    If lNum > 5 Then lNum = 5
    lblNumGroups.Caption = CStr(lNum)
    picFocus.SetFocus
    Call EnableDisableGroups

End Sub

Private Sub cmdViewCalc_Click(Index As Integer)

    Dim sTemp As String
    Dim dblTop As Double, dblBottom As Double
    Dim lTempScanFrom() As Long, lTempScanTo() As Long
    Dim Count As Long

    If Categories(gnCurCat).NumberOfGroups < 1 Then Exit Sub

    Load fmCalculations

    'Mu calculations
    ReDim lTempScanFrom(1 To Categories(gnCurCat).NumberOfGroups)
    ReDim lTempScanTo(1 To Categories(gnCurCat).NumberOfGroups)

    For Count = 1 To Categories(gnCurCat).NumberOfGroups
        lTempScanFrom(Count) = Categories(gnCurCat).Groups(Count).ScanFrom
        lTempScanTo(Count) = Categories(gnCurCat).Groups(Count).ScanTo
    Next Count

    'Display calculations
    With fmCalculations

```

```

'LB
.lblEq1.Caption = "Equation1: Calculation of LB" & (Index + 1)

.lblEqualsLB(0).Caption = "LB" & CStr(Index + 1) & " ="

.SumViewLB(0).ToValue = CStr(Categories(gnCurCat).NumberOfGroups)
.SumViewLB(1).ToValue = CStr(Categories(gnCurCat).NumberOfGroups)
.SumViewLB(3).FromValue = CStr(Index + 1)
.SumViewLB(3).ToValue = CStr(Categories(gnCurCat).NumberOfGroups)
.SumViewLB(4).FromValue = CStr(Index + 1)
.SumViewLB(4).ToValue = CStr(Categories(gnCurCat).NumberOfGroups)
.SumViewLB(5).FromValue = CStr(Index + 1)
.SumViewLB(5).ToValue = CStr(Index + 1) & "-1"

'Calculate top
dblTop = CalculateSumOf(CLng(Index + 1), Categories(gnCurCat).NumberOfGroups,
lTempScanFrom, lTempScanTo, True, sTemp)
.lblLB(6).Caption = sTemp
.lblLB(7).Caption = sTemp

'Calculate bottom-right (bottom-left = top)
dblBottom = CalculateSumOf(1, CLng(Index), lTempScanFrom, lTempScanTo, False,
sTemp)
.lblLB(8).Caption = sTemp

.lblLB(9).Caption = CStr(dblTop) & " / " & CStr(dblTop + dblBottom)
If Fix(Categories(gnCurCat).Groups(Index + 1).Mu_Lowerbound) <>
Categories(gnCurCat).Groups(Index + 1).Mu_Lowerbound Then
    .lblLB(10).Caption = Format(Categories(gnCurCat).Groups(Index +
1).Mu_Lowerbound, "0.###")
Else
    .lblLB(10).Caption = Categories(gnCurCat).Groups(Index + 1).Mu_Lowerbound
End If

'UB
.lblEq2.Caption = "Equation2: Calculation of UB" & (Index + 1)

.lblEqualsUB(0).Caption = "UB" & CStr(Index + 1) & " ="

.SumViewUB(0).ToValue = CStr(Categories(gnCurCat).NumberOfGroups)
.SumViewUB(1).ToValue = CStr(Categories(gnCurCat).NumberOfGroups)
.SumViewUB(3).FromValue = CStr(Index + 1)
.SumViewUB(3).ToValue = CStr(Categories(gnCurCat).NumberOfGroups)
.SumViewUB(4).FromValue = CStr(Index + 1)
.SumViewUB(4).ToValue = CStr(Categories(gnCurCat).NumberOfGroups)
.SumViewUB(5).FromValue = CStr(Index + 1)
.SumViewUB(5).ToValue = CStr(Index + 1) & "-1"

'Calculate top
dblTop = CalculateSumOf(CLng(Index + 1), Categories(gnCurCat).NumberOfGroups,
lTempScanFrom, lTempScanTo, False, sTemp)
.lblUB(6).Caption = sTemp
.lblUB(7).Caption = sTemp

'Calculate bottom-right (bottom-left = top)
dblBottom = CalculateSumOf(1, CLng(Index), lTempScanFrom, lTempScanTo, True,
sTemp)
.lblUB(8).Caption = sTemp

.lblUB(9).Caption = CStr(dblTop) & " / " & CStr(dblTop + dblBottom)
If Fix(Categories(gnCurCat).Groups(Index + 1).Mu_Upperbound) <>
Categories(gnCurCat).Groups(Index + 1).Mu_Upperbound Then
    .lblUB(10).Caption = Format(Categories(gnCurCat).Groups(Index +
1).Mu_Upperbound, "0.###")
Else
    .lblUB(10).Caption = Categories(gnCurCat).Groups(Index + 1).Mu_Upperbound
End If
End With

'display form
fmCalculations.Show vbModal

End Sub

Private Sub Command1_Click()
    picFocus.SetFocus
End Sub

```

```

Private Sub Command2_Click()
    picFocus.SetFocus
End Sub

Private Sub Form_Activate()
    Screen.MousePointer = vbNormal
    picFocus.SetFocus
End Sub

Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)
    btnDown = Shift
    If Not bCbOperatorsGotFocus Then
        Select Case KeyCode
            Case vbKeyDown: Call cmdDown_Click
            Case vbKeyUp: Call cmdUp_Click
        End Select
    End If
End Sub

Private Sub Form_Load()

    'Center form on screen
    Me.Left = (Screen.Width - Me.Width) / 2
    Me.Top = (Screen.Height - Me.Height) / 2

    Frame7.Caption = "Computation of " & Chr(181) & " value"
    Label21.Caption = Chr(181) & "s"
    Frame4.Caption = "Calculation of MIN(" & Chr(181) & "i, C(xi))"
    Label23.Caption = "MIN(" & Chr(181) & "i,C(xi))"
    Label24.Caption = "Calculation of MIN(" & Chr(181) & "i, C(xi))"

    nTabIndex = 1
    frmGraph.ZOrder 0
    bCbOperatorsGotFocus = False
    gbFmSetupLoaded = True

End Sub

Private Sub Form_Unload(Cancel As Integer)
    Call UnloadThisForm(False)
    gbFmSetupLoaded = False
End Sub

Private Sub UnloadThisForm(Optional bUnloadMe As Boolean = True)
    If gbFmGraphicsLoaded Then Unload fmGraphics
    If bUnloadMe Then Unload Me
End Sub

Private Sub frmWizard_DblClick(Index As Integer)

    Dim Count As Long

    If (Index = 0) Then
        If btnDown = (vbCtrlMask + vbShiftMask + vbAltMask) Then
            Select Case gnCurCat
                Case 1
                    cbOperatorLower(1).ListIndex = 1
                    lblValLower(1).Text = "10%"
                    cbOperatorLower(2).ListIndex = 1
                    lblValLower(2).Text = "10%"
                    cbOperatorLower(3).ListIndex = 0
                    lblValLower(3).Text = "1"
                    cbOperatorLower(4).ListIndex = 2
                    lblValLower(4).Text = "2"

                    cbOperatorUpper(1).ListIndex = 1
                    lblValUpper(1).Text = "1"
                    cbOperatorUpper(2).ListIndex = 0
                    lblValUpper(2).Text = "10%"
                    cbOperatorUpper(3).ListIndex = 0
                    lblValUpper(3).Text = "10%"
                    cbOperatorUpper(4).ListIndex = 4

                    lblNumGroups.Caption = 4
                    Call EnableDisableGroups
                End Select
            End If
        End If
    End If
End Sub

```

```

txtScanFrom(0).Text = "2"
txtScanTo(0).Text = "2"
cbAdjectives(0).ListIndex = -1
txtVulnFrom(0).Text = "30"
txtVulnTo(0).Text = "35"
txtScanFrom(1).Text = "6"
txtScanTo(1).Text = "6"
cbAdjectives(1).ListIndex = 1
txtVulnFrom(1).Text = "40"
txtVulnTo(1).Text = "40"
txtScanFrom(2).Text = "4"
txtScanTo(2).Text = "4"
cbAdjectives(2).ListIndex = 0
txtVulnFrom(2).Text = "50"
txtVulnTo(2).Text = "50"
txtScanFrom(3).Text = "3"
txtScanTo(3).Text = "3"
cbAdjectives(3).ListIndex = 2
txtVulnFrom(3).Text = "50"
txtVulnTo(3).Text = "50"

txtTo.Text = "60"
cbRule.ListIndex = 1
txtRule.Text = "40"

Case 2
cbOperatorLower(1).ListIndex = xMINUS
lblValLower(1).Text = "5%"
cbOperatorLower(2).ListIndex = xMINUS
lblValLower(2).Text = "5%"
cbOperatorLower(3).ListIndex = xPLUS
lblValLower(3).Text = "1"
cbOperatorLower(4).ListIndex = xMULTIPLY
lblValLower(4).Text = "2"

cbOperatorUpper(1).ListIndex = xMINUS
lblValUpper(1).Text = "1"
cbOperatorUpper(2).ListIndex = xPLUS
lblValUpper(2).Text = "5%"
cbOperatorUpper(3).ListIndex = xPLUS
lblValUpper(3).Text = "5%"
cbOperatorUpper(4).ListIndex = xINFINITY

lblNumGroups.Caption = 3
Call EnableDisableGroups

txtScanFrom(0).Text = "3"
txtScanTo(0).Text = "3"
cbAdjectives(0).ListIndex = -1
txtVulnFrom(0).Text = "0"
txtVulnTo(0).Text = "0"
txtScanFrom(1).Text = "3"
txtScanTo(1).Text = "3"
cbAdjectives(1).ListIndex = -1
txtVulnFrom(1).Text = "1"
txtVulnTo(1).Text = "3"
txtScanFrom(2).Text = "9"
txtScanTo(2).Text = "9"
cbAdjectives(2).ListIndex = 1
txtVulnFrom(2).Text = "5"
txtVulnTo(2).Text = "5"

txtTo.Text = "100"
txtDivider.Text = "10"
cbRule.ListIndex = 1
txtRule.Text = "5"

Case 3
cbOperatorLower(1).ListIndex = xMINUS
lblValLower(1).Text = "50%"
cbOperatorLower(2).ListIndex = xMINUS
lblValLower(2).Text = "50%"
cbOperatorLower(3).ListIndex = xPLUS
lblValLower(3).Text = "50%"
cbOperatorLower(4).ListIndex = xMULTIPLY
lblValLower(4).Text = "5"

```

```

        cbOperatorUpper(1).ListIndex = xMINUS
        lblValUpper(1).Text = "0"
        cbOperatorUpper(2).ListIndex = xPLUS
        lblValUpper(2).Text = "50%"
        cbOperatorUpper(3).ListIndex = xPLUS
        lblValUpper(3).Text = "50%"
        cbOperatorUpper(4).ListIndex = xINFINITY

        lblNumGroups.Caption = 2
        Call EnableDisableGroups

        txtScanFrom(0).Text = "12"
        txtScanTo(0).Text = "12"
        cbAdjectives(0).ListIndex = -1
        txtVulnFrom(0).Text = "0"
        txtVulnTo(0).Text = "0"
        txtScanFrom(1).Text = "2"
        txtScanTo(1).Text = "2"
        cbAdjectives(1).ListIndex = -1
        txtVulnFrom(1).Text = "1"
        txtVulnTo(1).Text = "1"

        txtTo.Text = "11"
        txtDivider.Text = "5"
        cbRule.ListIndex = 2
        txtRule.Text = "1"

    End Select

End If
End If

End Sub

Private Sub GraphView1_Click()

    Dim ColVals() As Long
    Dim lNumCols As Long

    Load fmGraphics

    With fmGraphics
        .GraphView1.Special_LineColor = Options.Prediction_LineColor
        .GraphView1.Prediction_LineColor = Options.Prediction_ColumnColor

        .GraphView1.Heading = GraphView1.Heading
        .GraphView1.XAxis_Heading = GraphView1.XAxis_Heading
        .GraphView1.XAxis_Increment = GraphView1.XAxis_Increment
        .GraphView1.XAxis_Max = GraphView1.XAxis_Max
        .GraphView1.XAxis_Min = GraphView1.XAxis_Min
        .GraphView1.YAxis_Heading = GraphView1.YAxis_Heading
        .GraphView1.YAxis_Increment = GraphView1.YAxis_Increment
        .GraphView1.YAxis_Max = GraphView1.YAxis_Max
        .GraphView1.YAxis_Min = GraphView1.YAxis_Min

        lNumCols = GraphView1.GetGraphColumnValues(ColVals)
        Call .GraphView1.DrawGraphColumns(ColVals)

        .GraphView1.Special_LineColor = GraphView1.Special_LineColor
        .GraphView1.Special_Lowerbound = GraphView1.Special_Lowerbound
        .GraphView1.Special_Upperbound = GraphView1.Special_Upperbound
        .GraphView1.Special_Display = GraphView1.Special_Display
        .GraphView1.XAxis_Values = ""
    End With

    picFocus.SetFocus
    fmGraphics.Show

End Sub

Public Sub SetupFormWithCategoryInfo(CatIndex As Integer)

    On Error GoTo ErrorHandler

    Dim Count As Long, lTemp As Long, CountIn As Long, lNum As Long
    Dim lNumAdj As Long
    Dim sngTop As Single

```



```

Label26.Caption = "IF the amount of vulnerabilities found for vulnerability category
" & gnCurCat & " is"

'Setup adjective info
lNumAdj = UBound(gsAdjectives)
If lNumAdj > 0 Then

'Setup adjective combos
cbRule.Clear
For Count = 0 To (cbAdjectives.Count - 1)
    cbAdjectives(Count).Clear
Next Count

sngTop = cbOperatorLower(0).Top
For Count = 1 To lNumAdj
    'Setup Lower Operator combo
    Load cbOperatorLower(Count)
    Call AddComboOperators(cbOperatorLower(Count))
    cbOperatorLower(Count).Top = sngTop
    sngTop = cbOperatorLower(Count).Top + cbOperatorLower(Count).Height + 15
    cbOperatorLower(Count).Visible = True

'Setup adjective
    Load lblAdjective(Count)
    lblAdjective(Count).Top = cbOperatorLower(Count).Top +
((cbOperatorLower(Count).Height - lblAdjective(Count).Height) / 2)
    lblAdjective(Count).Caption = gsAdjectives(Count)
    lblAdjective(Count).Visible = True

'Setup lower x-label
    Load lblXLower(Count)
    lblXLower(Count).Top = cbOperatorLower(Count).Top +
((cbOperatorLower(Count).Height - lblXLower(Count).Height) / 2)
    lblXLower(Count).Visible = True

'Setup lower value
    Load lblValLower(Count)
    lblValLower(Count).Top = cbOperatorLower(Count).Top
    lblValLower(Count).Visible = True

'Setup upper x-label
    Load lblXUpper(Count)
    lblXUpper(Count).Top = lblXLower(Count).Top
    lblXUpper(Count).Visible = True

'Setup Upper Operator combo
    Load cbOperatorUpper(Count)
    Call AddComboOperators(cbOperatorUpper(Count))
    cbOperatorUpper(Count).Top = cbOperatorLower(Count).Top
    cbOperatorUpper(Count).Visible = True

'Setup upper value
    Load lblValUpper(Count)
    lblValUpper(Count).Top = lblValLower(Count).Top
    lblValUpper(Count).Visible = True

'Add adjectives
For CountIn = 0 To (cbAdjectives.Count - 1)
    cbAdjectives(CountIn).AddItem gsAdjectives(Count)
    cbAdjectives(CountIn).ItemData(cbAdjectives(CountIn).NewIndex) = Count
Next CountIn
cbRule.AddItem gsAdjectives(Count)
cbRule.ItemData(cbRule.NewIndex) = Count

Next Count

'Check sizes
If (cbOperatorLower(lNumAdj).Top + cbOperatorLower(lNumAdj).Height) > (Shapel.Top
+ Shapel.Height) Then
    Shapel.Height = (cbOperatorLower(lNumAdj).Top + cbOperatorLower(lNumAdj).Height
+ 100)
    Frame2.Height = Shapel.Height + 270
    GraphView1.Height = Shapel.Height + 270
    Frame1.Top = Frame2.Top + Frame2.Height + 75
    cmdAccept.Top = Frame1.Top + Frame1.Height + 30
    cmdCancel.Top = cmdAccept.Top

```

```

    Me.Height = cmdAccept.Top + cmdAccept.Height + 423
End If

End If

If Categories(gnCurCat).HasBeenSetup Then
'Groups
lblNumGroups.Caption = Categories(gnCurCat).NumberOfGroups

For Count = 1 To Categories(gnCurCat).NumberOfGroups
txtScanFrom(Count - 1).Text = Categories(gnCurCat).Groups(Count).ScanFrom
txtScanTo(Count - 1).Text = Categories(gnCurCat).Groups(Count).ScanTo

    For CountIn = 1 To cbAdjectives(Count - 1).ListCount
        If cbAdjectives(Count - 1).List(CountIn - 1) =
Categories(gnCurCat).Groups(Count).Adjective Then
            cbAdjectives(Count - 1).ListIndex = CountIn - 1
            GoTo AfterForIn
        End If
    Next CountIn

AfterForIn:
    txtVulnFrom(Count - 1).Text =
Categories(gnCurCat).Groups(Count).VulnerabilityFrom
    txtVulnTo(Count - 1).Text = Categories(gnCurCat).Groups(Count).VulnerabilityTo
Next Count
Call EnabledisableGroups

'Adjectives
lNum = UBound(gsAdjectives)
For Count = 1 To lNum
    cbOperatorLower(Count).ListIndex =
Categories(gnCurCat).Adjectives(Count).LowerOperator
    lblValLower(Count).Text = Categories(gnCurCat).Adjectives(Count).LowerValue

    cbOperatorUpper(Count).ListIndex =
Categories(gnCurCat).Adjectives(Count).UpperOperator
    lblValUpper(Count).Text = Categories(gnCurCat).Adjectives(Count).UpperValue
Next Count

'Membership function
cbMFOps(0).ListIndex = Categories(gnCurCat).Membership_Op1
txtDevider.Text = Categories(gnCurCat).Membership_Devider
cbMFOps(1).ListIndex = Categories(gnCurCat).Membership_Op2
cbMFOps(2).ListIndex = Categories(gnCurCat).Membership_Op3
cbMFOps(3).ListIndex = Categories(gnCurCat).Membership_Op4
txtTo.Text = Categories(gnCurCat).Membership_To

'Rule
For Count = 0 To cbRule.ListCount - 1
    If cbRule.List(Count) = Categories(gnCurCat).Rule_Name Then
        cbRule.ListIndex = Count
        Exit For
    End If
Next Count
If cbRule.ListIndex < 0 Then cbRule.ListIndex = 0
txtRule.Text = Categories(gnCurCat).Rule_Value

Call DoTranslationTab

If TabStrip1.Tabs.Count < 2 Then
    TabStrip1.Tabs.Add 2, "Translation", "Translation"
    TabStrip1.Tabs.Add 3, "Forecast", "Forecast"
Else
    If TabStrip1.Tabs.Count < 3 Then
        TabStrip1.Tabs.Add 3, "Forecast", "Forecast"
    End If
End If
cmdBack.Visible = False
Else
lTemp = 10 - (Categories(gnCurCat).MaxVulnerabilityValue Mod 10)
lTemp = Categories(gnCurCat).MaxVulnerabilityValue + lTemp
'Membership function combos
cbMFOps(0).ListIndex = 3
txtDevider.Text = lTemp
cbMFOps(1).ListIndex = 1
cbMFOps(2).ListIndex = 1

```

```

    cbMFOps(3).ListIndex = 4
    txtTo.Text = lTemp
End If

Quit:
Exit Sub

ErrorHandler:
MsgBox "ERROR (SetupFormWithCategoryInfo)" & vbCrLf & Err.Number & ": " &
Err.Description, vbOKOnly + vbInformation, "Error Encountered"
Resume Quit

End Sub

```

```

Private Sub AddComboOperators(OperatorCombo As ComboBox)

    On Error Resume Next

    OperatorCombo.Clear
    OperatorCombo.AddItem "+"
    OperatorCombo.ItemData(OperatorCombo.NewIndex) = xPLUS
    OperatorCombo.AddItem "-"
    OperatorCombo.ItemData(OperatorCombo.NewIndex) = xMINUS
    OperatorCombo.AddItem "*"
    OperatorCombo.ItemData(OperatorCombo.NewIndex) = xMULTIPLY
    OperatorCombo.AddItem "/"
    OperatorCombo.ItemData(OperatorCombo.NewIndex) = xDEVIDE
    OperatorCombo.AddItem "inf"
    OperatorCombo.ItemData(OperatorCombo.NewIndex) = xINFINITY

End Sub

```

```

Private Sub EnableDisableGroups()

    Dim lNum As Long, Count As Long

    lNum = CLng(lblNumGroups.Caption)
    If (lNum < 1) Or (lNum > 5) Then Exit Sub

    'Disable groups not being used
    If lNum < 5 Then
        For Count = (lNum - 1) To 4
            lblGroup(Count).Enabled = False
            Label1(Count).Enabled = False
            txtScanFrom(Count).Enabled = False
            Label1(Count + 5).Enabled = False
            txtScanTo(Count).Enabled = False
            Label1(Count + 10).Enabled = False
            cbAdjectives(Count).Enabled = False
            txtVulnFrom(Count).Enabled = False
            Label1(Count + 15).Enabled = False
            txtVulnTo(Count).Enabled = False
            Label1(Count + 20).Enabled = False
        Next Count
    End If

    'Enable groups being used
    For Count = 0 To (lNum - 1)
        lblGroup(Count).Enabled = True
        Label1(Count).Enabled = True
        txtScanFrom(Count).Enabled = True
        Label1(Count + 5).Enabled = True
        txtScanTo(Count).Enabled = True
        Label1(Count + 10).Enabled = True
        cbAdjectives(Count).Enabled = True
        txtVulnFrom(Count).Enabled = True
        Label1(Count + 15).Enabled = True
        txtVulnTo(Count).Enabled = True
        Label1(Count + 20).Enabled = True
    Next Count

End Sub

```

```

Private Function CheckInfo() As Boolean

    On Error Resume Next

```



```

Dim bOK As Boolean
Dim Count As Long

bOK = True

Select Case nTabIndex
Case 1
    'Check adjectives
    For Count = 1 To (lblAdjective.Count - 1)
        If cbOperatorLower(Count).ListIndex < 0 Then
            If TabStrip1.SelectedItem.Index <> 1 Then TabStrip1.Tabs(1).Selected = True
            Call MsgBox("Please specify an operator (+, -, *, /)!", vbOKOnly +
vbInformation, "Check Info")
            cbOperatorLower(Count).SetFocus
            bOK = False
            GoTo Quit
        End If

        If cbOperatorLower(Count).ItemData(cbOperatorLower(Count).ListIndex) <>
xINFINITY Then
            If Not CheckIfContentsOfTextboxIsNumeric(lblValLower(Count)) Then
                If TabStrip1.SelectedItem.Index <> 1 Then TabStrip1.Tabs(1).Selected =
True
                Call MsgBox("Invalid numeric value!", vbOKOnly + vbInformation, "Check
Info")
                lblValLower(Count).SetFocus
                bOK = False
                GoTo Quit
            End If

            If cbOperatorUpper(Count).ListIndex < 0 Then
                If TabStrip1.SelectedItem.Index <> 1 Then TabStrip1.Tabs(1).Selected = True
                Call MsgBox("Please specify an operator (+, -, *, /)!", vbOKOnly +
vbInformation, "Check Info")
                cbOperatorUpper(Count).SetFocus
                bOK = False
                GoTo Quit
            End If

            If cbOperatorUpper(Count).ItemData(cbOperatorUpper(Count).ListIndex) <>
xINFINITY Then
                If Not CheckIfContentsOfTextboxIsNumeric(lblValUpper(Count)) Then
                    If TabStrip1.SelectedItem.Index <> 1 Then TabStrip1.Tabs(1).Selected =
True
                    Call MsgBox("Invalid numeric value!", vbOKOnly + vbInformation, "Check
Info")
                    lblValUpper(Count).SetFocus
                    bOK = False
                    GoTo Quit
                End If
            End If
        End If
    Next Count

    'Check groups
    For Count = 0 To (lblGroup.Count - 1)
        If lblGroup(Count).Enabled Then
            If Not CheckIfContentsOfTextboxIsNumeric(txtScanFrom(Count), False) Then
                If TabStrip1.SelectedItem.Index <> 1 Then TabStrip1.Tabs(1).Selected =
True
                Call MsgBox("Invalid numeric value!", vbOKOnly + vbInformation, "Check
Info")
                txtScanFrom(Count).SetFocus
                bOK = False
                GoTo Quit
            End If

            If Not CheckIfContentsOfTextboxIsNumeric(txtScanTo(Count), False) Then
                If TabStrip1.SelectedItem.Index <> 1 Then TabStrip1.Tabs(1).Selected =
True
                Call MsgBox("Invalid numeric value!", vbOKOnly + vbInformation, "Check
Info")
                txtScanTo(Count).SetFocus
                bOK = False
                GoTo Quit
            End If
        End If
    End If

```

```

        If Not CheckIfContentsOfTextboxIsNumeric(txtVulnFrom(Count), False) Then
        If TabStrip1.SelectedItem.Index <> 1 Then TabStrip1.Tabs(1).Selected =
True
        Call MsgBox("Invalid numeric value!", vbOKOnly + vbInformation, "Check
Info")
        txtVulnFrom(Count).SetFocus
        bOK = False
        GoTo Quit
        End If

        If Not CheckIfContentsOfTextboxIsNumeric(txtVulnTo(Count), False) Then
True
        Call MsgBox("Invalid numeric value!", vbOKOnly + vbInformation, "Check
Info")
        txtVulnTo(Count).SetFocus
        bOK = False
        GoTo Quit
        End If
    End If
Next Count

'Check Membership function
If Not CheckIfContentsOfTextboxIsNumeric(txtDevider, False) Then
    Call MsgBox("Invalid numeric value!", vbOKOnly + vbInformation, "Check Info")
    txtDevider.SetFocus
    bOK = False
    GoTo Quit
End If

If Not CheckIfContentsOfTextboxIsNumeric(txtTo, False) Then
    Call MsgBox("Invalid numeric value!", vbOKOnly + vbInformation, "Check Info")
    txtTo.SetFocus
    bOK = False
    GoTo Quit
End If

Categories(gnCurCat).Membership_Op1 = cbMFops(0).ListIndex
Categories(gnCurCat).Membership_Devider = CLng(txtDevider.Text)
Categories(gnCurCat).Membership_Op2 = cbMFops(1).ListIndex
Categories(gnCurCat).Membership_Op3 = cbMFops(2).ListIndex
Categories(gnCurCat).Membership_To = CLng(txtTo.Text)
Categories(gnCurCat).Membership_Op4 = cbMFops(3).ListIndex

Case 2
End Select

Quit:
    CheckInfo = bOK

End Function

Private Function CheckIfContentsOfTextboxIsNumeric(CheckTextbox As TextBox, Optional
bRightCharCanBePercentage As Boolean = True) As Boolean

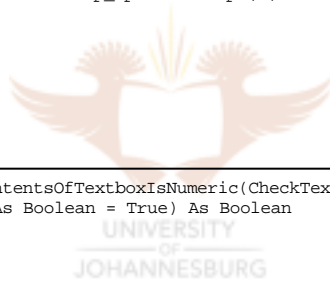
    Dim bOK As Boolean
    Dim sTemp As String

    bOK = True
    sTemp = CheckTextbox.Text
    If sTemp = "" Then
        bOK = False
        GoTo Quit
    End If

    If Not IsNumeric(sTemp) Then
        If bRightCharCanBePercentage Then
            If Right$(sTemp, 1) = "%" Then
                sTemp = Trim(Left$(sTemp, Len(sTemp) - 1))
                If Not IsNumeric(sTemp) Then bOK = False
            Else
                bOK = False
            End If
        Else
            bOK = False
        End If
    End If

End If

```



```

Quit:
    CheckIfContentsOfTextboxIsNumeric = bOK

End Function

```

```

Private Sub TabStrip1_Click()

    Dim Count As Long, lNum As Long

    If nTabIndex = TabStrip1.SelectedItem.Index Then Exit Sub

    lNum = Abs(nTabIndex - TabStrip1.SelectedItem.Index)
    If nTabIndex > TabStrip1.SelectedItem.Index Then
        If lNum > 1 Then
            For Count = 1 To (lNum - 1)
                Call cmdBackClick(False)
            Next Count
            Call cmdBackClick
        Else
            Call cmdBackClick
        End If
    Else
        If lNum > 1 Then
            For Count = 1 To (lNum - 1)
                Call cmdAcceptClick(False)
            Next Count
            Call cmdAcceptClick
        Else
            Call cmdAcceptClick
        End If
    End If

    nTabIndex = TabStrip1.SelectedItem.Index
End Sub

```

```

Private Function TranslateAdjectives(CategoryIndex As Long, GroupIndex As Long,
Optional NewFrom As Double = 0, Optional NewTo As Double = 0) As String

    On Error Resume Next

    Dim Count As Long, lTemp As Long, lNum As Long
    Dim sTemp As String, sVal As String, sChar As String
    Dim nIndex As Integer

    sTemp = ""
    nIndex = 0
    If Categories(CategoryIndex).Groups(GroupIndex).Adjective = "" Then
        NewFrom = Categories(CategoryIndex).Groups(GroupIndex).VulnerabilityFrom
        NewTo = Categories(CategoryIndex).Groups(GroupIndex).VulnerabilityTo
        sTemp = Categories(CategoryIndex).Groups(GroupIndex).VulnerabilityFrom & " to " &
Categories(CategoryIndex).Groups(GroupIndex).VulnerabilityTo
    Else
        lNum = UBound(gsAdjectives)
        For Count = 1 To lNum
            If gsAdjectives(Count) = Categories(CategoryIndex).Groups(GroupIndex).Adjective
Then
                nIndex = Count
                Exit For
            End If
        Next Count

        'From value
        sVal = Categories(CategoryIndex).Adjectives(nIndex).LowerValue
        If Right$(sVal, 1) = "%" Then
            sVal = Left(sVal, Len(sVal) - 1)
            lTemp = CLng(sVal)

            lTemp = (Categories(CategoryIndex).Groups(GroupIndex).VulnerabilityFrom * lTemp)
/ 100
        Else
            lTemp = CLng(sVal)
        End If

        Select Case Categories(CategoryIndex).Adjectives(nIndex).LowerOperator
            Case xPLUS: NewFrom =
Categories(CategoryIndex).Groups(GroupIndex).VulnerabilityFrom + lTemp

```

```

    Case xMINUS: NewFrom =
Categories(CategoryIndex).Groups(GroupIndex).VulnerabilityFrom - lTemp
    Case xMULTIPLY: NewFrom =
Categories(CategoryIndex).Groups(GroupIndex).VulnerabilityFrom * lTemp
    Case xDEVIDE: NewFrom =
Categories(CategoryIndex).Groups(GroupIndex).VulnerabilityFrom / lTemp
    Case xINFINITY: NewFrom = -1
End Select

If NewFrom = -1 Then
    sTemp = "INF to "
Else
    If NewFrom <> Fix(NewFrom) Then
        sTemp = Format(NewFrom, "0.###") & " to "
    Else
        sTemp = NewFrom & " to "
    End If
End If

'To value
sVal = Categories(CategoryIndex).Adjectives(nIndex).UpperValue
If Right$(sVal, 1) = "%" Then
    sVal = Left(sVal, Len(sVal) - 1)
    lTemp = CLng(sVal)

    lTemp = (Categories(CategoryIndex).Groups(GroupIndex).VulnerabilityTo * lTemp) /
100
Else
    lTemp = CLng(sVal)
End If

Select Case Categories(CategoryIndex).Adjectives(nIndex).UpperOperator
    Case xPLUS: NewTo = Categories(CategoryIndex).Groups(GroupIndex).VulnerabilityTo
+ lTemp
    Case xMINUS: NewTo =
Categories(CategoryIndex).Groups(GroupIndex).VulnerabilityTo - lTemp
    Case xMULTIPLY: NewTo =
Categories(CategoryIndex).Groups(GroupIndex).VulnerabilityTo * lTemp
    Case xDEVIDE: NewTo =
Categories(CategoryIndex).Groups(GroupIndex).VulnerabilityTo / lTemp
    Case xINFINITY: NewTo = -1
End Select

If NewTo = -1 Then
    sTemp = sTemp & "INF"
Else
    If NewTo <> Fix(NewTo) Then
        sTemp = sTemp & Format(NewTo, "0.###")
    Else
        sTemp = sTemp & NewTo
    End If
End If

TranslateAdjectives = sTemp
End Function

```

```

Private Function TranslateSpecificAdjective(LowerOperator As Long, LowerValue As
String, UpperOperator As Long, UpperValue As String, lValueToTranslate As Double,
uValueToTranslate As Double) As String

    On Error Resume Next

    Dim Count As Long, lTemp As Double, lNum As Long
    Dim sTemp As String, sVal As String, sChar As String

    sTemp = ""

    'From value
    sVal = LowerValue
    If Right$(sVal, 1) = "%" Then
        sVal = Left(sVal, Len(sVal) - 1)
        lTemp = CLng(sVal)

        lTemp = (lValueToTranslate * lTemp) / 100
    Else

```



```

    lTemp = CLng(sVal)
End If

Select Case LowerOperator
    Case xPLUS: lValueToTranslate = lValueToTranslate + lTemp
    Case xMINUS: lValueToTranslate = lValueToTranslate - lTemp
    Case xMULTIPLY: lValueToTranslate = lValueToTranslate * lTemp
    Case xDEVIDE: lValueToTranslate = lValueToTranslate / lTemp
    'Case xINFINITY: NewFrom = -1
End Select

'To value
sVal = UpperValue
If Right$(sVal, 1) = "%" Then
    sVal = Left(sVal, Len(sVal) - 1)
    lTemp = CLng(sVal)

    lTemp = (uValueToTranslate * lTemp) / 100
Else
    lTemp = CLng(sVal)
End If

Select Case UpperOperator
    Case xPLUS: uValueToTranslate = uValueToTranslate + lTemp
    Case xMINUS: uValueToTranslate = uValueToTranslate - lTemp
    Case xMULTIPLY: uValueToTranslate = uValueToTranslate * lTemp
    Case xDEVIDE: uValueToTranslate = uValueToTranslate / lTemp
    'Case xINFINITY: NewTo = -1
End Select

sTemp = RangeFormatForDbl(lValueToTranslate, uValueToTranslate)
TranslateSpecificAdjective = sTemp

End Function

```

```

Private Function ComputeCharFunction(CategoryIndex As Long, GroupIndex As Long,
Optional FromValue As Double = -1, Optional ToValue As Double = -1) As String

    Dim lRule As Long
    Dim sTemp As String

    'From value
    sTemp = ""
    lRule = GetRuleNumber(Categories(CategoryIndex),
Categories(CategoryIndex).Groups(GroupIndex).VulnerabilityTranslatedFrom)
    Select Case lRule
        Case 1: FromValue = 0
        Case 2: FromValue =
(Categories(CategoryIndex).Groups(GroupIndex).VulnerabilityTranslatedFrom /
Categories(CategoryIndex).Membership_Devider)
        Case 3: FromValue = 1
    End Select

    'To value
    lRule = GetRuleNumber(Categories(CategoryIndex),
Categories(CategoryIndex).Groups(GroupIndex).VulnerabilityTranslatedTo)
    Select Case lRule
        Case 1: ToValue = 0
        Case 2: ToValue =
(Categories(CategoryIndex).Groups(GroupIndex).VulnerabilityTranslatedTo /
Categories(CategoryIndex).Membership_Devider)
        Case 3: ToValue = 1
    End Select

    sTemp = RangeFormatForDbl(FromValue, ToValue)
    ComputeCharFunction = sTemp

End Function

```

```

Private Function GetRuleNumber(Membership As CategoryType, dblFromToValue As Double)
As Long

    Dim lRule As Long, lTemp As Long

    lRule = -1
    Select Case Membership.Membership_Op1
        Case xEQUALS: If dblFromToValue = 0 Then lRule = 1
    End Select

```



```

Case xLESS_THAN: If dblFromToValue < 0 Then lRule = 1
Case xGREATER_THAN: If dblFromToValue > 0 Then lRule = 1
Case xLESS_THAN_OR_EQUAL: If dblFromToValue <= 0 Then lRule = 1
Case xGREATER_THAN_OR_EQUAL: If dblFromToValue >= 0 Then lRule = 1
End Select

If lRule = -1 Then
  Select Case Membership.Membership_Op2
  Case xEQUALS
    If (dblFromToValue = 0) Then
      lTemp = Membership.Membership_To
      Select Case Membership.Membership_Op3
      Case xEQUALS: If (dblFromToValue = lTemp) Then lRule = 2
      Case xLESS_THAN: If (dblFromToValue < lTemp) Then lRule = 2
      Case xGREATER_THAN: If (dblFromToValue > lTemp) Then lRule = 2
      Case xLESS_THAN_OR_EQUAL: If (dblFromToValue <= lTemp) Then lRule = 2
      Case xGREATER_THAN_OR_EQUAL: If (dblFromToValue >= lTemp) Then lRule = 2
      End Select
    End If

  Case xLESS_THAN
    If (0 < dblFromToValue) Then
      lTemp = Membership.Membership_To
      Select Case Membership.Membership_Op3
      Case xEQUALS: If (dblFromToValue = lTemp) Then lRule = 2
      Case xLESS_THAN: If (dblFromToValue < lTemp) Then lRule = 2
      Case xGREATER_THAN: If (dblFromToValue > lTemp) Then lRule = 2
      Case xLESS_THAN_OR_EQUAL: If (dblFromToValue <= lTemp) Then lRule = 2
      Case xGREATER_THAN_OR_EQUAL: If (dblFromToValue >= lTemp) Then lRule = 2
      End Select
    End If

  Case xGREATER_THAN
    If (0 > dblFromToValue) Then
      lTemp = Membership.Membership_To
      Select Case Membership.Membership_Op3
      Case xEQUALS: If (dblFromToValue = lTemp) Then lRule = 2
      Case xLESS_THAN: If (dblFromToValue < lTemp) Then lRule = 2
      Case xGREATER_THAN: If (dblFromToValue > lTemp) Then lRule = 2
      Case xLESS_THAN_OR_EQUAL: If (dblFromToValue <= lTemp) Then lRule = 2
      Case xGREATER_THAN_OR_EQUAL: If (dblFromToValue >= lTemp) Then lRule = 2
      End Select
    End If

  Case xLESS_THAN_OR_EQUAL
    If (0 <= dblFromToValue) Then
      lTemp = Membership.Membership_To
      Select Case Membership.Membership_Op3
      Case xEQUALS: If (dblFromToValue = lTemp) Then lRule = 2
      Case xLESS_THAN: If (dblFromToValue < lTemp) Then lRule = 2
      Case xGREATER_THAN: If (dblFromToValue > lTemp) Then lRule = 2
      Case xLESS_THAN_OR_EQUAL: If (dblFromToValue <= lTemp) Then lRule = 2
      Case xGREATER_THAN_OR_EQUAL: If (dblFromToValue >= lTemp) Then lRule = 2
      End Select
    End If

  Case xGREATER_THAN_OR_EQUAL
    If (0 >= dblFromToValue) Then
      lTemp = Membership.Membership_To
      Select Case Membership.Membership_Op3
      Case xEQUALS: If (dblFromToValue = lTemp) Then lRule = 2
      Case xLESS_THAN: If (dblFromToValue < lTemp) Then lRule = 2
      Case xGREATER_THAN: If (dblFromToValue > lTemp) Then lRule = 2
      Case xLESS_THAN_OR_EQUAL: If (dblFromToValue <= lTemp) Then lRule = 2
      Case xGREATER_THAN_OR_EQUAL: If (dblFromToValue >= lTemp) Then lRule = 2
      End Select
    End If

  End Select
Else
  If lRule = -1 Then lRule = 3
End If

GetRuleNumber = lRule

End Function

```

```

Private Sub DoTranslationTab()

    On Error GoTo ErrorHandler

    Dim Count As Long, lNum As Long, lIndex As Long
    Dim lTempScanFrom() As Long, lTempScanTo() As Long
    Dim dblTopSum As Double, dblBottomSumR As Double, dblMaxL As Double, dblMaxU As
    Double
    Dim sTemp As String, sConcl As String, sMAX As String
    Dim s1 As String, s2 As String

    lNum = 0
    For Count = 1 To Categories(gnCurCat).NumberOfGroups
        'Translate
        lblTranslGroup(Count - 1).Caption = "In " &
        Categories(gnCurCat).Groups(Count).ScanFrom & " to " &
        Categories(gnCurCat).Groups(Count).ScanTo & " scan(s)"
        lblTranslGroup(Count - 1).Visible = True
        lblTransVulns(Count - 1).Caption = TranslateAdjectives(CLng(gnCurCat), Count,
        Categories(gnCurCat).Groups(Count).VulnerabilityTranslatedFrom,
        Categories(gnCurCat).Groups(Count).VulnerabilityTranslatedTo)
        lblTransVulns(Count - 1).Visible = True

        'Char function
        lblDistrGroup(Count - 1).Caption =
        RangeFormatForDbl(CDbl(Categories(gnCurCat).Groups(Count).ScanFrom),
        CDbl(Categories(gnCurCat).Groups(Count).ScanTo))
        lblDistrGroup(Count - 1).Visible = True

        lblDistrVulns(Count - 1).Caption = ComputeCharFunction(CLng(gnCurCat), Count,
        Categories(gnCurCat).Groups(Count).Cx_From, Categories(gnCurCat).Groups(Count).Cx_To)
        lblDistrVulns(Count - 1).Visible = True

        lblVulns(Count - 1).Caption = lblDistrVulns(Count - 1).Caption
        lblVulns(Count - 1).Visible = True
        cmdViewCalc(Count - 1).Visible = True

        lblMINGroup(Count - 1).Visible = True
        lblMINResult(Count - 1).Visible = True
    Next Count

    If Categories(gnCurCat).NumberOfGroups < 5 Then
        For Count = (Categories(gnCurCat).NumberOfGroups + 1) To 5
            lblTranslGroup(Count - 1).Visible = False
            lblTransVulns(Count - 1).Visible = False
            lblDistrGroup(Count - 1).Visible = False
            lblDistrVulns(Count - 1).Visible = False

            lblVulns(Count - 1).Visible = False
            lblMu(Count - 1).Visible = False
            cmdViewCalc(Count - 1).Visible = False

            lblMINGroup(Count - 1).Visible = False
            lblMINResult(Count - 1).Visible = False
        Next Count
    End If

    'Mu calculations
    ReDim lTempScanFrom(1 To Categories(gnCurCat).NumberOfGroups)
    ReDim lTempScanTo(1 To Categories(gnCurCat).NumberOfGroups)

    For Count = 1 To Categories(gnCurCat).NumberOfGroups
        lTempScanFrom(Count) = Categories(gnCurCat).Groups(Count).ScanFrom
        lTempScanTo(Count) = Categories(gnCurCat).Groups(Count).ScanTo
    Next Count

    sConcl = ""
    For Count = 1 To Categories(gnCurCat).NumberOfGroups

        'Calculate lower bound
        dblTopSum = CalculateSumOf(Count, Categories(gnCurCat).NumberOfGroups,
        lTempScanFrom, lTempScanTo)
        dblBottomSumR = CalculateSumOf(1, Count - 1, lTempScanFrom, lTempScanTo, False)
        Categories(gnCurCat).Groups(Count).Mu_Lowerbound = dblTopSum / (dblTopSum +
        dblBottomSumR)

        'Calculate upper bound

```

```

    dblTopSum = CalculateSumOf(Count, Categories(gnCurCat).NumberOfGroups,
lTempScanFrom, lTempScanTo, False)
    dblBottomSumR = CalculateSumOf(1, Count - 1, lTempScanFrom, lTempScanTo)
    Categories(gnCurCat).Groups(Count).Mu_Upperbound = dblTopSum / (dblTopSum +
dblBottomSumR)

    sTemp = RangeFormatForDbl(Categories(gnCurCat).Groups(Count).Mu_Lowerbound,
Categories(gnCurCat).Groups(Count).Mu_Upperbound)
    lblMu(Count - 1).Caption = sTemp

    'Display min mu and C(x)
    lblMINGroup(Count - 1).Caption = "MIN(" & lblMu(Count - 1).Caption & ", " &
lblVulns(Count - 1).Caption & ")"

    'Calculate MIN(mu,C(x))
    sTemp = MinMuGamma(Categories(gnCurCat).Groups(Count).Mu_Lowerbound,
Categories(gnCurCat).Groups(Count).Mu_Upperbound,
Categories(gnCurCat).Groups(Count).Cx_From, Categories(gnCurCat).Groups(Count).Cx_To,
Categories(gnCurCat).Groups(Count).MIN_Cx_Mu_LB,
Categories(gnCurCat).Groups(Count).MIN_Cx_Mu_UB)
    lblMINResult(Count - 1).Caption = sTemp

    'Forecast
    If sConcl = "" Then
        sConcl = sTemp
    Else
        sConcl = sConcl & ", " & sTemp
    End If

    'Calculate Maximum of (minimums of mu and C(x))
    If Count = 1 Then
        Categories(gnCurCat).MAXofMIN_Cx_Mu.Lowerbound =
Categories(gnCurCat).Groups(Count).MIN_Cx_Mu_LB
        Categories(gnCurCat).MAXofMIN_Cx_Mu.Upperbound =
Categories(gnCurCat).Groups(Count).MIN_Cx_Mu_UB
        sMAX = RangeFormatForDbl(Categories(gnCurCat).MAXofMIN_Cx_Mu.Lowerbound,
Categories(gnCurCat).MAXofMIN_Cx_Mu.Upperbound)
    Else
        sMAX = MaxMuGamma(Categories(gnCurCat).MAXofMIN_Cx_Mu.Lowerbound,
Categories(gnCurCat).MAXofMIN_Cx_Mu.Upperbound,
Categories(gnCurCat).Groups(Count).MIN_Cx_Mu_LB,
Categories(gnCurCat).Groups(Count).MIN_Cx_Mu_UB,
Categories(gnCurCat).MAXofMIN_Cx_Mu.Lowerbound,
Categories(gnCurCat).MAXofMIN_Cx_Mu.Upperbound)
    End If

    Next Count

    'Display Forecast
    lblConcl(0).Caption = "MAX(MIN(" & Chr(181) & "i, C(xi))"
    lblConcl(1).Caption = "= MAX(" & sConcl & ")"
    lblConcl(2).Caption = "= " & sMAX

    lblMore(0).Caption = "The amount of vulnerabilities found for vulnerability category
" & gnCurCat & " = " & cbRule.Text & " " & sMAX & " * " &
Categories(gnCurCat).Membership_Devider

    If cbRule.Text <> "" Then

        For Count = 1 To 6
            lblMore(Count).Visible = True
        Next Count

        lblMore(1).Caption = "Get rid of " & cbRule.Text & " part:"
        lblMore(2).Caption = "" & cbRule.Text & " " & sMAX

        lNum = -1
        lIndex = -1
        lNum = UBound(gsAdjectives)
        For Count = 1 To lNum
            If LCase(gsAdjectives(Count)) = LCase(cbRule.Text) Then
                lIndex = Count
            Exit For
        End If
        Next Count

        If lIndex > 0 Then

```

```

    If Fix(Categories(gnCurCat).MAXofMIN_Cx_Mu.Lowerbound) =
Categories(gnCurCat).MAXofMIN_Cx_Mu.Lowerbound Then
    sTemp = "[" & CStr(Categories(gnCurCat).MAXofMIN_Cx_Mu.Lowerbound)
Else
    sTemp = "[" & Format(Categories(gnCurCat).MAXofMIN_Cx_Mu.Lowerbound, "0.0##")
End If
Select Case Categories(gnCurCat).Adjectives(lIndex).LowerOperator
Case xPLUS: sTemp = sTemp & " + "
Case xMINUS: sTemp = sTemp & " - "
Case xMULTIPLY: sTemp = sTemp & " * "
Case xDEVIDE: sTemp = sTemp & " / "
Case xINFINITY
End Select
If Fix(Categories(gnCurCat).MAXofMIN_Cx_Mu.Upperbound) =
Categories(gnCurCat).MAXofMIN_Cx_Mu.Upperbound Then
    sTemp = sTemp & Categories(gnCurCat).Adjectives(lIndex).LowerValue & ", " &
CStr(Categories(gnCurCat).MAXofMIN_Cx_Mu.Upperbound)
Else
    sTemp = sTemp & Categories(gnCurCat).Adjectives(lIndex).LowerValue & ", " &
Format(Categories(gnCurCat).MAXofMIN_Cx_Mu.Upperbound, "0.0##")
End If

Select Case Categories(gnCurCat).Adjectives(lIndex).UpperOperator
Case xPLUS: sTemp = sTemp & " + "
Case xMINUS: sTemp = sTemp & " - "
Case xMULTIPLY: sTemp = sTemp & " * "
Case xDEVIDE: sTemp = sTemp & " / "
Case xINFINITY
End Select
sTemp = sTemp & Categories(gnCurCat).Adjectives(lIndex).UpperValue & "]"
lblMore(4).Caption = sTemp

Categories(gnCurCat).Final.Lowerbound =
Categories(gnCurCat).MAXofMIN_Cx_Mu.Lowerbound '0.6
Categories(gnCurCat).Final.Upperbound =
Categories(gnCurCat).MAXofMIN_Cx_Mu.Upperbound '0.733
sTemp =
TranslateSpecificAdjective(Categories(gnCurCat).Adjectives(lIndex).LowerOperator,
Categories(gnCurCat).Adjectives(lIndex).LowerValue,
Categories(gnCurCat).Adjectives(lIndex).UpperOperator,
Categories(gnCurCat).Adjectives(lIndex).UpperValue,
Categories(gnCurCat).Final.Lowerbound, Categories(gnCurCat).Final.Upperbound)
'1 10% 0 10%
lblMore(6).Caption = sTemp

lblMore(7).Caption = sTemp & " * " & Categories(gnCurCat).Membership_Devider

Categories(gnCurCat).Final_AmtVulns.Lowerbound =
Round(Categories(gnCurCat).Final.Lowerbound * Categories(gnCurCat).Membership_Devider)
Categories(gnCurCat).Final_AmtVulns.Upperbound =
Round(Categories(gnCurCat).Final.Upperbound * Categories(gnCurCat).Membership_Devider)
lblFinal.Caption =
RangeFormatForDbl(CDbl(Categories(gnCurCat).Final_AmtVulns.Lowerbound),
CDbl(Categories(gnCurCat).Final_AmtVulns.Upperbound))

sTemp = "The expected amount of vulnerabilities likely to be found in the future
for vulnerability category "
sTemp = sTemp & gnCurCat & ", is between " &
Categories(gnCurCat).Final_AmtVulns.Lowerbound & " and "
sTemp = sTemp & Categories(gnCurCat).Final_AmtVulns.Upperbound & "."
lblMore(9).Caption = sTemp
End If
Else
For Count = 1 To 6
    lblMore(Count).Visible = False
Next Count

lblMore(7).Caption = sMAX & " * " & Categories(gnCurCat).Membership_Devider
Categories(gnCurCat).Final_AmtVulns.Lowerbound =
Round(Categories(gnCurCat).Final.Lowerbound * Categories(gnCurCat).Membership_Devider)
Categories(gnCurCat).Final_AmtVulns.Upperbound =
Round(Categories(gnCurCat).Final.Upperbound * Categories(gnCurCat).Membership_Devider)
lblFinal.Caption =
RangeFormatForDbl(CDbl(Categories(gnCurCat).Final_AmtVulns.Lowerbound),
CDbl(Categories(gnCurCat).Final_AmtVulns.Upperbound))
End If

```

```

Quit:
Exit Sub

ErrorHandler:
If lNum = -1 Then
    lNum = 0
    Resume Next
End If
Resume Quit
Resume

End Sub

```

```

' This function will put x and y into the [x, y] range format
Function RangeFormatForDbl(LeftOperand As Double, RightOperand As Double) As String
    If Fix(LeftOperand) = LeftOperand Then
        If Fix(RightOperand) = RightOperand Then
            RangeFormatForDbl = "[" & CStr(LeftOperand) & ", " & CStr(RightOperand) & "]"
        Else
            RangeFormatForDbl = "[" & CStr(LeftOperand) & ", " & Format(RightOperand,
"0.0##") & "]"
        End If
    Else
        If Fix(LeftOperand) = LeftOperand Then
            RangeFormatForDbl = "[" & Format(LeftOperand, "0.0##") & ", " &
CStr(RightOperand) & "]"
        Else
            RangeFormatForDbl = "[" & Format(LeftOperand, "0.0##") & ", " &
Format(RightOperand, "0.0##") & "]"
        End If
    End If
End Function

```

```

' This function calculates the minimum range between each Mu and Gamma range
Private Function MinMuGamma(Mu1 As Double, Mu2 As Double, Gamma1 As Double, Gamma2 As
Double, Optional MinMuGamma1 As Double, Optional MinMuGamma2 As Double) As String

    Dim RangesSubsetIndicator As Integer

    If (((Gamma1 < Mu1) And (Gamma2 < Mu1)) Or ((Mu1 < Gamma1) And (Mu2 < Gamma1))) Then
        ' Mu and Gamma ranges don't intersect
        ' Apply Theorem 2
        If Gamma2 < Mu1 Then
            MinMuGamma1 = Gamma1
            MinMuGamma2 = Gamma2
            MinMuGamma = RangeFormatForDbl(Gamma1, Gamma2)
        Else
            ' Mu2 < Gamma1
            MinMuGamma1 = Mu1
            MinMuGamma2 = Mu2
            MinMuGamma = RangeFormatForDbl(Mu1, Mu2)
        End If
    Else
        ' Mu and Gamma ranges intersect
        RangesSubsetIndicator = RangesSubset(Mu1, Mu2, Gamma1, Gamma2)
        Select Case RangesSubsetIndicator
            Case 0 ' Neither range is a subset of the other
                ' Apply Theorem 4
                If Gamma2 < Mu2 Then
                    MinMuGamma1 = Gamma1
                    MinMuGamma2 = Gamma2
                    MinMuGamma = RangeFormatForDbl(Gamma1, Gamma2)
                Else
                    ' Mu2 < Gamma2
                    MinMuGamma1 = Mu1
                    MinMuGamma2 = Mu2
                    MinMuGamma = RangeFormatForDbl(Mu1, Mu2)
                End If
            Case 1 ' Ranges of Gamma is Subset of Mu
                MinMuGamma1 = Mu1
                MinMuGamma2 = Gamma2
                MinMuGamma = RangeFormatForDbl(Mu1, Gamma2)
            Case 2 ' Ranges of Mu is Subset of Gamma
                MinMuGamma1 = Gamma1
                MinMuGamma2 = Mu2

```

```

        MinMuGamma = RangeFormatForDbl(Gamma1, Mu2)
    End Select
End If
End Function

```

```

'This function calculates the maximum range between each Mu and Gamma range
Private Function MaxMuGamma(Mu1 As Double, Mu2 As Double, Gamma1 As Double, Gamma2 As
Double, Optional MaxMuGamma1 As Double, Optional MaxMuGamma2 As Double) As String

    Dim RangesSubsetIndicator As Integer

    If (((Gamma1 < Mu1) And (Gamma2 < Mu1)) Or ((Mu1 < Gamma1) And (Mu2 < Gamma1))) Then
        'Mu and Gamma ranges don't intersect
        'Apply Theorem 1
        If Gamma1 > Mu2 Then
            MaxMuGamma1 = Gamma1
            MaxMuGamma2 = Gamma2
            MaxMuGamma = RangeFormatForDbl(Gamma1, Gamma2)
        Else
            If Mu1 > Gamma2 Then
                MaxMuGamma1 = Mu1
                MaxMuGamma2 = Mu2
                MaxMuGamma = RangeFormatForDbl(Mu1, Mu2)
            End If
        End If
    Else
        'Mu and Gamma ranges intersect
        RangesSubsetIndicator = RangesSubset(Mu1, Mu2, Gamma1, Gamma2)
        Select Case RangesSubsetIndicator
            Case 0 'Neither range is a subset of the other
                'Apply Theorem 3
                If Gamma2 > Mu2 Then
                    MaxMuGamma1 = Gamma1
                    MaxMuGamma2 = Gamma2
                    MaxMuGamma = RangeFormatForDbl(Gamma1, Gamma2)
                Else
                    'Mu2 > Gamma2
                    MaxMuGamma1 = Mu1
                    MaxMuGamma2 = Mu2
                    MaxMuGamma = RangeFormatForDbl(Mu1, Mu2)
                End If

                Case 1 'Ranges of Gamma is Subset of Mu
                    MaxMuGamma1 = Gamma1
                    MaxMuGamma2 = Mu2
                    MaxMuGamma = RangeFormatForDbl(Gamma1, Mu2)

                Case 2 'Ranges of Mu is Subset of Gamma
                    MaxMuGamma1 = Mu1
                    MaxMuGamma2 = Gamma2
                    MaxMuGamma = RangeFormatForDbl(Mu1, Gamma2)
            End Select
    End Select
End If
End Function

```

```

'This function determines if ranges are subsets
Function RangesSubset(Mu1 As Double, Mu2 As Double, Gamma1 As Double, Gamma2 As
Double) As Integer
    RangesSubset = 0 'Neither range is a subset of the other
    If (Gamma1 >= Mu1) And (Gamma1 <= Mu2) And (Gamma2 >= Mu1) And (Gamma2 <= Mu2) Then
        'Ranges of Gamma is Subset of Mu
        RangesSubset = 1
    Else
        If (Mu1 >= Gamma1) And (Mu1 <= Gamma2) And (Mu2 >= Gamma1) And (Mu2 <= Gamma2)
Then
            'Ranges of Mu is Subset of Gamma
            RangesSubset = 2
        End If
    End If
End Function

```

```

Private Function CalculateSumOf(j As Long, n As Long, ScanValuesFrom() As Long,
ScanValuesTo() As Long, Optional bMIN As Boolean = True, Optional sSumString As String
= "") As Double

```

```

Dim Count As Long, lTemp As Long
Dim dblSum As Double

dblSum = 0
sSumString = ""
For Count = j To n
    If bMIN Then
        If ScanValuesFrom(Count) <= ScanValuesTo(Count) Then
            lTemp = ScanValuesFrom(Count)
        Else
            lTemp = ScanValuesTo(Count)
        End If
    Else
        If ScanValuesFrom(Count) >= ScanValuesTo(Count) Then
            lTemp = ScanValuesFrom(Count)
        Else
            lTemp = ScanValuesTo(Count)
        End If
    End If

    If sSumString = "" Then
        sSumString = CStr(lTemp)
    Else
        sSumString = sSumString & " + " & CStr(lTemp)
    End If

    dblSum = dblSum + lTemp
Next Count

If sSumString = "" Then sSumString = "0"
CalculateSumOf = dblSum

End Function

```

```

Private Sub txtTo_Change()
    Label14.Caption = txtTo.Text
End Sub

```

B.1.9 The “frmSetupNames” form

The design of this form is shown in figure B.10. This form is used to specify where the vulnerability database is accessed for the specific VS product used. In addition, the exact database tables and fields to be used are specified here.

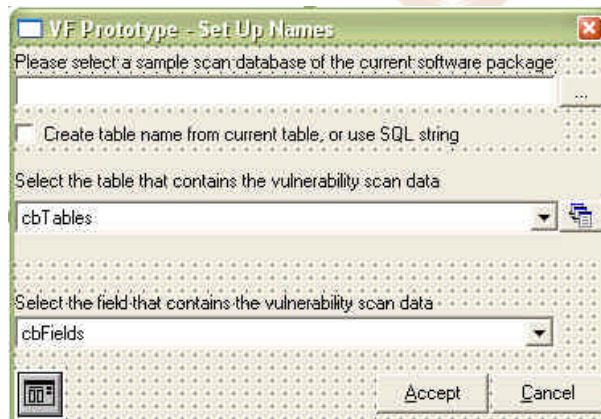


Figure B.10: The “frmSetupNames” form

The source code for this form follows below.

```

Option Explicit

Private Sub cbSQLTables_Click()
    Call LoadTableFields(True)
End Sub

Private Sub cbTables_Click()
    If chkSQL.Value = vbUnchecked Then Call LoadTableFields(True)
End Sub

Private Sub chkSQL_Click()
    frmWhat(0).Visible = Not (chkSQL.Value = vbChecked)
    frmWhat(1).Visible = (chkSQL.Value = vbChecked)
    cbFields.Clear
    cbFields.Visible = False
    Label2.Visible = False
End Sub

Private Sub cmdAccept_Click()

    Dim bOK As Boolean

    bOK = CheckInfo
    If bOK Then

        If gbFmOptionsLoaded Then
            If fmOptions.Visible Then
                If chkSQL.Value = vbChecked Then
                    fmOptions.txtScanTableName.Text = txtSQL.Text
                Else
                    fmOptions.txtScanTableName.Text = cbTables.Text
                End If
                fmOptions.txtScanFieldName.Text = cbFields.Text
            End If
        End If

        ModalResult = True
        Unload Me

    End If

End Sub

Private Sub cmdCancel_Click()
    ModalResult = False
    Unload Me
End Sub

Private Sub cmdHelp_Click()
    txtSQL.Text = "<REPLACE(<DB_FIELDVALUE(tablename.fieldname)>," & Chr(34) & "{" &
Chr(34) & "|" & Chr(34) & Chr(34) & "," & Chr(34) & "}" & Chr(34) & "|" & Chr(34) &
Chr(34) & "," & Chr(34) & "-" & Chr(34) & "|" & Chr(34) & "-" & Chr(34) &
")>_05050000_2"
End Sub

Private Sub cmdLoadTables_Click()

    On Error Resume Next

    Dim TempCat As New ADOX.Catalog
    Dim TempConn As New ADODB.Connection
    Dim Count As Long

    Screen.MousePointer = vbHourglass

    'Open connection
    TempConn.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" &
txtSWDir.Text & ";Persist Security Info=False"
    TempConn.Open

    'Open catalog
    TempCat.ActiveConnection = TempConn

```



```

cbTables.Clear
For Count = 0 To (TempCat.Tables.Count - 1)
  If TempCat.Tables(Count).Type = "TABLE" Then
    cbTables.AddItem TempCat.Tables(Count).Name
  End If
Next Count
If cbTables.ListCount > 0 Then
  cbTables.ListIndex = 0
  Call LoadTableFields
End If

Screen.MousePointer = vbNormal

End Sub

```

```

Private Sub cmdSelectDB_Click()

  On Error GoTo ErrorHandler

  Dim sDef As String

  cmnDlgSWDB.CancelError = True
  cmnDlgSWDB.DialogTitle = "Select Database"

  sDef = txtSWDir.Text
  If (sDef = "") Or (Len(sDef) < 2) Then sDef = App.Path

  cmnDlgSWDB.InitDir = sDef
  cmnDlgSWDB.ShowOpen

  If cmnDlgSWDB.FileName <> "" Then
    txtSWDir.Text = cmnDlgSWDB.FileName
  End If

Quit:
  Exit Sub

ErrorHandler:
  Resume Quit

End Sub

```

```

Private Sub cmdSpecial_Click()

  Dim TempConn As New ADODB.Connection
  Dim TempRS As New ADODB.Recordset
  Dim Count As Long, Place1 As Long, Place2 As Long
  Dim SQL As String, sTemp As String, sTable As String, sField As String
  Dim sReplaceString1 As String, sReplaceString2 As String
  Dim sReplaceWhat As String, sReplaceWith As String
  Dim lLine As Long

  If txtSQL.Text = "" Then Exit Sub

  If (InStr(1, LCase(txtSQL.Text), "tablename") > 0) Or (InStr(1, LCase(txtSQL.Text),
"fieldname") > 0) Then
    Call MsgBox("You have to replace 'tablename' with a valid table name and
'fieldname' with a valid field name from the table!", vbOKOnly + vbInformation)
    If txtSQL.Visible Then txtSQL.SetFocus
    Exit Sub
  End If

  Screen.MousePointer = vbHourglass

  sTemp = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & txtSWDir.Text & ";Persist
Security Info=False"

  sTemp = GetTableNameFromReplaceString(txtSQL.Text, sTemp)

  cbSQLTables.Clear
  If sTemp <> "" Then
    cbSQLTables.AddItem sTemp
    cbSQLTables.ListIndex = 0
    If sTemp <> "" Then
      cmdSpecialFields.Enabled = True
      cbSQLTables.Enabled = True
    End If
  End If

```

```

    End If
End If

Quit:
Screen.MousePointer = vbNormal
Exit Sub

ErrorHandler:
If lLine = 1 Then
    Call MsgBox("The specified table and/or field does not exist in the database",
vbOKOnly + vbInformation)
    Resume Quit
End If
Resume Quit

End Sub

Private Sub cmdSpecialFields_Click()
    Call LoadTableFields(True)
End Sub

Private Sub Form_Load()

    'Center form on screen
    Me.Left = (Screen.Width - Me.Width) / 2
    Me.Top = (Screen.Height - Me.Height) / 2

    Call SetupForm

    gbFmSetupNames = True

End Sub

Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
    If UnloadMode = vbManual Then
        ModalResult = False
    End If
End Sub

Private Sub Form_Unload(Cancel As Integer)
    gbFmSetupNames = False
End Sub

Private Sub txtSQL_Change()
    cmdSpecialFields.Enabled = False
    cbSQLTables.Enabled = False
End Sub

Private Sub LoadTableFields(Optional bDoMousePointer As Boolean = False)

    Dim TempConn As New ADODB.Connection
    Dim TempRS As New ADODB.Recordset
    Dim Count As Long
    Dim myCombo As ComboBox

    If chkSQL.Value = vbChecked Then
        Set myCombo = cbSQLTables
    Else
        Set myCombo = cbTables
    End If

    If myCombo.Text = "" Then Exit Sub

    If bDoMousePointer Then Screen.MousePointer = vbHourglass

    'Open connection
    TempConn.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" &
txtSwdDir.Text & ";Persist Security Info=False"
    TempConn.Open

    'Open table
    TempRS.CursorLocation = adUseServer
    TempRS.CursorType = adOpenStatic
    TempRS.LockType = adLockOptimistic
    TempRS.Open myCombo.Text, TempConn, adCmdTable

    cbFields.Clear

```

```

For Count = 0 To (TempRS.Fields.Count - 1)
    cbFields.AddItem TempRS.Fields(Count).Name
Next Count
If cbFields.ListCount > 0 Then
    cbFields.ListIndex = 0
    Label2.Visible = True
    cbFields.Visible = True
End If

Quit:
If bDoMousePointer Then Screen.MousePointer = vbNormal
Exit Sub

ErrorHandler:
Resume Quit

End Sub

```

```

Public Function GetTableNameFromReplaceString(sString As String,
sConnectionStringToDatabase As String) As String

    On Error GoTo ErrorHandler

    Dim TempConn As New ADODB.Connection
    Dim TempRS As New ADODB.Recordset
    Dim Count As Long, Place1 As Long, Place2 As Long
    Dim SQL As String, sTemp As String, sTable As String, sField As String
    Dim sReplaceString1 As String, sReplaceString2 As String
    Dim sReplaceWhat As String, sReplaceWith As String
    Dim lLine As Long

    If sString = "" Then Exit Function

    'Open connection
    TempConn.ConnectionString = sConnectionStringToDatabase
    TempConn.Open

    'Open table
    TempRS.CursorLocation = adUseServer
    TempRS.CursorType = adOpenStatic
    TempRS.LockType = adLockOptimistic

    sTemp = sString
    If InStr(1, LCase(sTemp), "<replace(") Then
        '<REPLACE(<DB_FIELDVALUE(dbo_Events.ProductID)>,"{|","|"}|","-|"_)>_05050000_2
        Place1 = InStr(1, sTemp, "<")
        Place2 = InStrRev(sTemp, ">")

        If (Place1 < 1) Or (Place2 < 1) Then
            SQL = sTemp
            GoTo Quit
        End If

        sReplaceString1 = Mid$(sTemp, Place1 + 1, Place2 - (Place1 + 1))
        sReplaceString1 = Replace(sReplaceString1, "REPLACE(", "") 'Remove REPLACE(
        sReplaceString1 = Left$(sReplaceString1, Len(sReplaceString1) - 1) 'Remove rH )

        Place1 = InStr(1, sReplaceString1, ",")
        If Place1 < 1 Then Exit Function

        sReplaceString2 = Left$(sReplaceString1, Place1 - 1)
        sReplaceString1 = Mid$(sReplaceString1, Place1 + 1)

        'Get table (and field?)
        Place1 = InStr(1, sReplaceString2, "<DB_FIELDVALUE(")
        If Place1 > 0 Then
            'Get value from the field in database
            sReplaceString2 = Replace(sReplaceString2, "<DB_FIELDVALUE(", "")
            sReplaceString2 = Left$(sReplaceString2, Len(sReplaceString2) - 2) 'Remove )>

            Place2 = InStr(1, sReplaceString2, ".") 'eg table.field
            If Place2 > 0 Then
                sTable = Left$(sReplaceString2, Place2 - 1)
                sField = Mid$(sReplaceString2, Place2 + 1)
            Else
                Screen.MousePointer = vbNormal
            End If
        End If
    End If

```

```

        Call MsgBox("You have to specify a table and field to retrieve the value
from!", vbOKOnly + vbInformation)
        Exit Function
    End If

    'Open recordset to get field
    lLine = 1
    TempRS.Open "SELECT TOP 1 " & sField & " FROM " & sTable, TempConn, adCmdTable
    lLine = 2
    If Not TempRS.EOF Then
        sTable = "" & TempRS.Fields(sField).Value
    End If

    TempRS.Close
    TempConn.Close
Else
    sTable = sReplaceString2
End If

If (sReplaceString1 = "") Or (sTable = "") Then Exit Function
Do While sReplaceString1 <> ""
    Place1 = InStr(1, sReplaceString1, ",")
    If Place1 > 0 Then
        sReplaceString2 = Left$(sReplaceString1, Place1 - 1)
        sReplaceString1 = Mid$(sReplaceString1, Place1 + 1)
    Else
        sReplaceString2 = sReplaceString1
        sReplaceString1 = ""
    End If

    Place2 = InStr(1, sReplaceString2, "|")
    If Place2 > 0 Then
        sReplaceWhat = Left$(sReplaceString2, Place2 - 1)
        sReplaceWhat = Replace(sReplaceWhat, Chr(34), "")
        sReplaceWth = Mid$(sReplaceString2, Place2 + 1)
        sReplaceWth = Replace(sReplaceWth, Chr(34), "")

        sTable = Replace(sTable, sReplaceWhat, sReplaceWth)
    End If
Loop

Else
    SQL = sTemp
End If

'Replace new table in original string
Place1 = InStr(1, LCase(sTemp), "<replace(")
Place2 = InStrRev(LCase(sTemp), ">")
If (Place1 > 0) And (Place2 > 0) Then
    sReplaceWhat = Mid$(sTemp, Place1, Place2 - Place1 + 1)
    sTemp = Replace(sTemp, sReplaceWhat, sTable)
End If

Quit:
GetTableNameFromReplaceString = sTemp
Exit Function

ErrorHandler:
If lLine = 1 Then
    Call MsgBox("The specified table and/or field does not exist in the database",
vbOKOnly + vbInformation)
    Resume Quit
End If
Resume Quit

End Function

```

```

Private Sub SetupForm()

    Dim nIndex As Integer
    Dim Count As Long

    txtSWDir.Text = SWPackageCurrent.ScanningDB.SampleDB.DBPathName

    If txtSWDir.Text <> "" Then
        chkSQL.Value = SWPackageCurrent.ScanningDB.SampleDB.CreatedFrom
    End If
End Sub

```

```

If chkSQL.Value = vbChecked Then
    txtSQL.Text = SWPackageCurrent.ScanningDB.TableName

    'load tables
    Call cmdSpecial_Click

    'Set up table
    nIndex = -1
    For Count = 0 To (cbSQLTables.ListCount - 1)
        If SWPackageCurrent.ScanningDB.SampleDB.SQLResultTableName =
cbSQLTables.List(Count) Then
            nIndex = Count
            Exit For
        End If
    Next Count
    cbSQLTables.ListIndex = nIndex

    'Load fields
    Call cmdSpecialFields_Click
Else
    'Load tables
    Call cmdLoadTables_Click

    'Set up table
    nIndex = -1
    For Count = 0 To (cbTables.ListCount - 1)
        If SWPackageCurrent.ScanningDB.TableName = cbTables.List(Count) Then
            nIndex = Count
            Exit For
        End If
    Next Count
    cbTables.ListIndex = nIndex

    'Load fields
    Call LoadTableFields

End If

'Set up field
nIndex = -1
For Count = 0 To (cbFields.ListCount - 1)
    If SWPackageCurrent.ScanningDB.FieldName = cbFields.List(Count) Then
        nIndex = Count
        Exit For
    End If
Next Count
cbFields.ListIndex = nIndex

Else
    chkSQL.Value = vbUnchecked
    frmWhat(1).Visible = False
    frmWhat(0).Visible = True
    cbTables.Clear
    Label2.Visible = False
    cbFields.Visible = False
End If

End Sub

Private Function CheckInfo() As Boolean

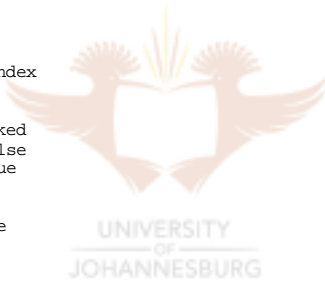
    Dim bOK As Boolean

    bOK = True

    'Check db path
    If (txtSWSDir.Text = "") Then
        bOK = False
        Call MsgBox("Please specify a valid database!", vbOKOnly + vbInformation)
        txtSWSDir.SetFocus
        GoTo Quit
    End If

    If chkSQL.Value = vbChecked Then
        'Check SQL
        If txtSQL.Text = "" Then
            bOK = False

```



```

        Call MsgBox("Please specify a valid SQL string to return a table!", vbOKOnly +
vbInformation)
        txtSQL.SetFocus
        GoTo Quit
    End If

    'Check tables
    If (cbSQLTables.Text = "") Then
        bOK = False
        Call MsgBox("Please specify a valid table!", vbOKOnly + vbInformation)
        cbSQLTables.SetFocus
        GoTo Quit
    End If
Else
    'Check tables
    If (cbTables.Text = "") Then
        bOK = False
        Call MsgBox("Please specify a valid table from the database!", vbOKOnly +
vbInformation)
        cbTables.SetFocus
        GoTo Quit
    End If
End If

'Check Field
If (cbFields.Text = "") Then
    bOK = False
    Call MsgBox("Please specify a valid field from the table!", vbOKOnly +
vbInformation)
    cbFields.SetFocus
    GoTo Quit
End If

'Set up info
With SWPackageCurrent.ScanningDB
    .SampleDB.DBPathName = txtSWDir.Text

    .SampleDB.CreatedFrom = CLng(chkSQL.Value)
    If chkSQL.Value = vbChecked Then
        .TableName = txtSQL.Text
        .SampleDB.SQLResultTableName = cbSQLTables.Text
    Else
        .TableName = cbTables.Text
    End If
    .FieldName = cbFields.Text
End With

Quit:
    CheckInfo = bOK
End Function

```

B.1.10 The “frmSWSetup” form

The design of this form is shown in figure B.11. This form is used to specify software-specific setup settings for the specific VS product used. This is an important form since the mapping of vulnerabilities onto the harmonised vulnerability categories is done for the VS product.

Figure B.11: The “frmSWSetup” form

The source code for this form follows below.

```

Option Explicit
Dim isActive As Boolean
Dim lScrollingValue As Long
Dim lButton As Long

Private Sub cbTables_Click()
    Call LoadTableFields
End Sub

Private Sub cmdAccept_Click()

    Dim bOK As Boolean

    bOK = CheckInfo
    If bOK Then
        If (LCase(fmOptions.cbVulnSWList.Text) = "<none>") Then
            fmOptions.cbVulnSWList.Clear
            fmOptions.cbVulnSWList.AddItem txtName.Text
            fmOptions.cbVulnSWList.ItemData(fmOptions.cbVulnSWList.NewIndex) =
                UBound(SWPackageInfo)

            If fmOptions.cbVulnSWList.ListCount > 0 Then fmOptions.cbVulnSWList.ListIndex = 0

            If SWPackageCurrent.Number = 0 Then

```

```

        SWPackageCurrent.Number = GetMaxID("SWPackage", "Package_ID") + 1
    End If
    Call AddSWPackageToDB(SWPackageCurrent)

    ModalResult = True
    Unload Me
End If

End Sub

```

```

Private Sub AddSWPackageToDB(SWPackage As SWPackageInfoType)

    On Error Resume Next

    Dim TempConn As New ADODB.Connection
    Dim TempRS As New ADODB.Recordset
    Dim TempRSSeq As New ADODB.Recordset
    Dim Count As Long, CountSeq As Long
    Dim sTemp As String
    Dim sSplit() As String

    Screen.MousePointer = vbHourglass

    'Open connection
    TempConn.ConnectionString = gsConnectionStringToMainDB
    TempConn.Open

    'Open recordset
    Call OpenDBTable(TempConn, TempRS, "SELECT * FROM SWPackage WHERE Package_ID=" &
SWPackage.Number)
    If TempRS.EOF Then
        TempRS.AddNew
    End If
    TempRS!Package_ID = SWPackage.Number
    TempRS!Package_Name = "" & SWPackage.Name
    TempRS!Vuln_DB_PathName = "" & SWPackage.VulnDB.DBPathName
    TempRS!Vuln_DB_TableName = "" & SWPackage.VulnDB.TableName
    TempRS!Vuln_DB_FieldName_ID = "" & SWPackage.VulnDB.FieldName_ID
    TempRS!Vuln_DB_FieldName_Description = "" & SWPackage.VulnDB.FieldName_Description
    TempRS!Scan_DB_TableName = "" & SWPackage.ScanningDB.TableName
    TempRS!Scan_DB_FieldName = "" & SWPackage.ScanningDB.FieldName
    TempRS.Update

    'Close recordset
    TempRS.Close

    '*****
    'Write categories to DB
    '*****
    TempConn.Execute "DELETE FROM Vulnerability"
    TempConn.Execute "DELETE FROM VulnerabilityCategory"

    'Open recordset
    Call OpenDBTable(TempConn, TempRS, "SELECT * FROM VulnerabilityCategory")
    Call OpenDBTable(TempConn, TempRSSeq, "SELECT * FROM Vulnerability")
    For Count = 1 To lvMainCats.ListItems.Count
        TempRS.AddNew
        TempRS!VulnerabilityCategory_Number = SWPackage.VulnDB.MainCategories(Count).ID
    '1000
        TempRS!VulnerabilityCategory_ID = SWPackage.VulnDB.MainCategories(Count).Number
    '1
        TempRS!VulnerabilityCategory_Description = "" &
SWPackage.VulnDB.MainCategories(Count).Description
        TempRS!InternalPackageID =
fmMain.cbSWPackage.ItemData(fmMain.cbSWPackage.ListIndex)
        TempRS.Update

        'Add vulns for this category
        sTemp = SWPackage.VulnDB.MainCategories(Count).Sequence
        Call CreateCommaSeperatedNumbersFromOptimizedString(sTemp)
        sSplit = Split(sTemp, ",")
        For CountSeq = 0 To UBound(sSplit)
            TempRSSeq.AddNew
            TempRSSeq!Vulnerability_Number = CLng(sSplit(CountSeq))
            TempRSSeq!Vulnerability_Description =
FindVulnerabilityDescription(CLng(sSplit(CountSeq)))

```



```

        TempRSSeq!VulnerabilityCategory_Number =
SWPackage.VulnDB.MainCategories(Count).ID
        TempRSSeq.Update
        Next CountSeq

Next Count

'Close recordsets
TempRSSeq.Close
TempRS.Close

'*****
'Write mapping to DB
'*****
TempConn.Execute "DELETE FROM Mapping"
'Open recordset
Call OpenDBTable(TempConn, TempRS, "SELECT * FROM Mapping")
For Count = 1 To (txtMap.Count - 1)
    sTemp = txtMap(Count).Text
    Call CreateCommaSeperatedNumbersFromOptimizedString(sTemp)
    sSplit = Split(sTemp, ",")
    For CountSeq = 0 To UBound(sSplit)
        TempRS.AddNew
            TempRS!HVC_Number = Count
            TempRS!Vulnerability_Number = CLng(sSplit(CountSeq))
            TempRS!InternalPackageID =
fmMain.cbSWPackage.ItemData(fmMain.cbSWPackage.ListIndex)
            TempRS.Update
            Next CountSeq
        Next Count

TempRS.Close

'Close connection
TempConn.Close

Screen.MousePointer = vbNormal

End Sub

```

```

Private Function FindVulnerabilityDescription(VulnNumber As Long) As String

    Dim Count As Long
    Dim sTemp As String

    On Error GoTo ErrorHandler

    sTemp = ""
    For Count = 1 To lvSWCategories.ListItems.Count
        If CLng(lvSWCategories.ListItems(Count).Text) = VulnNumber Then
            sTemp = lvSWCategories.ListItems(Count).SubItems(1)
            Exit For
        End If
    Next Count

Quit:
    FindVulnerabilityDescription = sTemp
    Exit Function

ErrorHandler:
    sTemp = ""
    Resume Quit

End Function

```

```

Private Sub cmdCancel_Click()
    ModalResult = False
    Unload Me
End Sub

```

```

Private Sub cmdCatsIn_Click()

    On Error GoTo Quit

    Dim lNumSelected As Long
    Dim Count As Long, CountIn As Long
    Dim sTemp As String, sString As String, sF As String, sI As String

```

```

Dim sSplit() As String
Dim PlaceS As Long, PlaceNextS As Long, lFirstNumberInSequence As Long,
lNextNumberInSequence As Long

If lvMainCats.ListItems.Count < 1 Then Exit Sub

lNumSelected = 0
For Count = 1 To lvMainCats.ListItems.Count
    If lvMainCats.ListItems(Count).Selected Then
        lNumSelected = lNumSelected + 1
    End If
Next Count

If lNumSelected < 1 Then Exit Sub

With fmSelectCats.lvSWCategories

    sTemp = lvMainCats.SelectedItem.SubItems(3)
    Call CreateCommaSeperatedNumbersFromOptimizedString(sTemp)
    If sTemp <> "" Then
        sSplit = Split(sTemp, ",")
    Else
        ReDim sSplit(0)
    End If

    For Count = 1 To .ListItems.Count
        .ListItems(Count).Checked = False
        .ListItems(Count).Selected = False

        For CountIn = 0 To UBound(sSplit)
            If sSplit(CountIn) = .ListItems(Count).Text Then
                .ListItems(Count).Checked = True
                GoTo AfterCountIn
            End If
        Next CountIn

    AfterCountIn:
        Next Count

    If .ListItems.Count > 0 Then
        .ListItems(1).Selected = True
        .SelectedItem.EnsureVisible
    End If

End With

fmSelectCats.txtNumber.Text = lvMainCats.SelectedItem.Text
fmSelectCats.txtNumber.SelStart = 0
fmSelectCats.txtNumber.SelLength = Len(fmSelectCats.txtNumber.Text)

fmSelectCats.Show vbModal

Quit:
Exit Sub

End Sub

Private Sub cmdLoadCatInfo_Click()

    On Error GoTo ErrorHandler

    Dim TempConn As New ADODB.Connection
    Dim TempRS As New ADODB.Recordset
    Dim Count As Long
    Dim li As ListItem
    Dim sTemp As String

    Screen.MousePointer = vbHourglass

    'Open connection
    TempConn.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" &
    txtSWDir.Text & ";Persist Security Info=False"
    TempConn.Open

    'Open recordset
    TempRS.CursorLocation = adUseServer
    TempRS.CursorType = adOpenStatic

```

```

TempRS.LockType = adLockOptimistic
TempRS.Open "SELECT * FROM " & cbTables.Text & " WHERE " & cbID.Text & " > 0 ORDER
BY " & cbID.Text & " ASC", TempConn, adCmdTable

lvSWCategories.ListItems.Clear
Do While Not TempRS.EOF
    Set li = lvSWCategories.ListItems.Add(, "Key" & TempRS.Fields(cbID.Text).Value,
TempRS.Fields(cbID.Text).Value)
    sTemp = TempRS.Fields(cbDescription.Text).Value
    sTemp = Replace(sTemp, vbCrLf, "")
    sTemp = Replace(sTemp, vbLf, "")
    sTemp = Replace(sTemp, vbCr, "")
    li.SubItems(1) = sTemp

    Set li = fmSelectCats.lvSWCategories.ListItems.Add(, "Key" &
TempRS.Fields(cbID.Text).Value, TempRS.Fields(cbID.Text).Value)
    sTemp = TempRS.Fields(cbDescription.Text).Value
    sTemp = Replace(sTemp, vbCrLf, "")
    sTemp = Replace(sTemp, vbLf, "")
    sTemp = Replace(sTemp, vbCr, "")
    li.SubItems(1) = sTemp

    TempRS.MoveNext
Loop

'Close recordset and connection
TempRS.Close
TempConn.Close

frmMappingInfo.Visible = True
If lvSWCategories.ListItems.Count > 0 Then
    lvSWCategories.ListItems(1).Selected = True

    frmMappingInfo.Visible = True
    If txtMap(1).Visible Then txtMap(1).SetFocus
End If

Quit:
Screen.MousePointer = vbNormal
Exit Sub

ErrorHandler:
Resume Quit

End Sub

Private Sub cmdLoadTables_Click()

    On Error Resume Next

    Dim TempCat As New ADOX.Catalog
    Dim TempConn As New ADODB.Connection
    Dim Count As Long

    Screen.MousePointer = vbHourglass

    'Open connection
    TempConn.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" &
txtSwdDir.Text & ";Persist Security Info=False"
    TempConn.Open

    'Open catalog
    TempCat.ActiveConnection = TempConn

    cbTables.Clear
    For Count = 0 To (TempCat.Tables.Count - 1)
        If TempCat.Tables(Count).Type = "TABLE" Then
            cbTables.AddItem TempCat.Tables(Count).Name
        End If
    Next Count
    If cbTables.ListCount > 0 Then
        cbTables.ListIndex = 0
        Call LoadTableFields
        cmdLoadCatInfo.Enabled = True
        Label5.Visible = True
        lvExample.Visible = True
        Label2.Visible = True
    End If

```

```

    cbID.Visible = True
    Label4.Visible = True
    cbDescription.Visible = True
    cmdLoadCatInfo.Visible = True
End If

Screen.MousePointer = vbNormal

End Sub

```

```

Private Sub cmdMap_Click()

    On Error GoTo ErrorHandler

    Dim sTemp As String
    Dim nIndex As Integer
    Dim Count As Long, CountIn As Long
    Dim li As ListItem

    If frmMain(1).Visible Then
        For Count = 1 To (lblNum.Count - 1)
            If lblNum(Count).FontBold Then
                nIndex = Cint(Count)
                Exit For
            End If
        Next Count

        For Count = 1 To lvSWCategories.ListItems.Count
            If lvSWCategories.ListItems(Count).Selected Then
                sTemp = lvSWCategories.ListItems(Count)
                If IsNumeric(sTemp) Then
                    If txtMap(nIndex).Text = "" Then
                        txtMap(nIndex).Text = sTemp
                    Else
                        txtMap(nIndex).Text = txtMap(nIndex).Text & "," & sTemp
                    End If
                End If
            End If
        Next Count
    Else
        For Count = 1 To lvSWCategories.ListItems.Count
            If lvSWCategories.ListItems(Count).Selected Then
                sTemp = lvSWCategories.ListItems(Count)
                If IsNumeric(sTemp) Then
                    Call AddMainCategory(CLng(lvSWCategories.ListItems(Count).Text),
lvSWCategories.ListItems(Count).SubItems(1), lvSWCategories.ListItems(Count).Key)
                End If
            End If
        BeforeNext:
        Next Count

        Call ReorderMainCategories
    End If

    Exit Sub

ErrorHandler:
    If Err.Number = 35602 Then
        'Item already exists
        GoTo BeforeNext
    End If
    Resume Next

End Sub

```

```

Private Sub AddMainCategory(Number As Long, sDescription As String, sKey As String)

    Dim TempArray() As CustomCategoryInfoType
    Dim Count As Long, lNumCats As Long, lTempIndex As Long
    Dim bAdded As Boolean
    Dim li As ListItem

    lNumCats = lvMainCats.ListItems.Count
    ReDim TempArray(1 To lNumCats + 1)

    If lNumCats = 0 Then
        TempArray(1).Number = Number
    End If

```

```

TempArray(1).ID = Number
TempArray(1).Description = sDescription
TempArray(1).Key = sKey
TempArray(1).Sequence = ""
Else
lTempIndex = 0
bAdded = False
For Count = 1 To lNumCats
    If Number = CLng(lvMainCats.ListItems(Count).Text) Then Exit Sub

    lTempIndex = lTempIndex + 1
    If Number > CLng(lvMainCats.ListItems(Count).Text) Then
        TempArray(lTempIndex).Number = CLng(lvMainCats.ListItems(Count).Text)
        TempArray(lTempIndex).ID = CLng(lvMainCats.ListItems(Count).SubItems(1))
        TempArray(lTempIndex).Description = lvMainCats.ListItems(Count).SubItems(2)
        TempArray(lTempIndex).Key = lvMainCats.ListItems(Count).Key
        TempArray(lTempIndex).Sequence = lvMainCats.ListItems(Count).SubItems(3)
    Else
        If Not bAdded Then
            TempArray(lTempIndex).Number = Number
            TempArray(lTempIndex).ID = Number
            TempArray(lTempIndex).Description = sDescription
            TempArray(lTempIndex).Key = sKey
            TempArray(lTempIndex).Sequence = ""
            Count = Count - 1
            bAdded = True
        Else
            TempArray(lTempIndex).Number = CLng(lvMainCats.ListItems(Count).Text)
            TempArray(lTempIndex).ID = CLng(lvMainCats.ListItems(Count).SubItems(1))
            TempArray(lTempIndex).Description = lvMainCats.ListItems(Count).SubItems(2)
            TempArray(lTempIndex).Key = lvMainCats.ListItems(Count).Key
            TempArray(lTempIndex).Sequence = lvMainCats.ListItems(Count).SubItems(3)
        End If
    End If
Next Count

If Not bAdded Then
    TempArray(lNumCats + 1).Number = Number
    TempArray(lNumCats + 1).ID = Number
    TempArray(lNumCats + 1).Description = sDescription
    TempArray(lNumCats + 1).Key = sKey
    TempArray(lNumCats + 1).Sequence = ""
End If
End If

lvMainCats.ListItems.Clear
For Count = 1 To (lNumCats + 1)
    Set li = lvMainCats.ListItems.Add(, TempArray(Count).Key, TempArray(Count).Number)
    li.SubItems(1) = TempArray(Count).ID
    li.SubItems(2) = TempArray(Count).Description
    li.SubItems(3) = TempArray(Count).Sequence
Next Count
End Sub

```

```

Private Sub cmdOptimize_Click()

    Dim Count As Long
    Dim sTemp As String

    For Count = 1 To (txtMap.Count - 1)
        sTemp = txtMap(Count).Text
        Call OptimizeCommaSeperatedNumbers(sTemp)
        txtMap(Count).Text = sTemp
    Next Count
End Sub

```

```

Private Sub cmdRemove_Click()

    Dim lNumSelected As Long, Count As Long

    If lvMainCats.ListItems.Count < 1 Then Exit Sub

    lNumSelected = 0
    For Count = 1 To lvMainCats.ListItems.Count
        If lvMainCats.ListItems(Count).Selected Then

```

```

        lNumSelected = lNumSelected + 1
    End If
Next Count

If lNumSelected < 1 Then Exit Sub

Call lvMainCats.ListItems.Remove(lvMainCats.SelectedItem.Index)
If lvMainCats.ListItems.Count > 0 Then
    lvMainCats.ListItems(1).Selected = True
    Call lvMainCats.ListItems(1).EnsureVisible
End If

End Sub

Private Sub cmdSelectDBDir_Click()

    On Error GoTo ErrorHandler

    Dim sDef As String

    cmnDlgSWDB.CancelError = True
    cmnDlgSWDB.DialogTitle = "Select Database"

    sDef = txtSWDir.Text
    If (sDef = "") Or (Len(sDef) < 2) Then sDef = App.Path

    cmnDlgSWDB.InitDir = sDef
    cmnDlgSWDB.ShowOpen

    If cmnDlgSWDB.FileName <> "" Then
        txtSWDir.Text = cmnDlgSWDB.FileName
    End If

Quit:
    Exit Sub

ErrorHandler:
    Resume Quit

End Sub

Private Sub Form_Activate()

    If Not isActive Then
        If txtName.Visible Then
            txtName.SelStart = 0
            txtName.SelLength = Len(txtName.Text)
            txtName.SetFocus
        End If
        isActive = True
    End If

End Sub

Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)
    lButton = Shift
End Sub

Private Sub Form_Load()

    isActive = False

    'Center form on screen
    Me.Left = (Screen.Width - Me.Width) / 2
    Me.Top = (Screen.Height - Me.Height) / 2

    Call SetupForm
    Load fmSelectCats

    lScrollingValue = 150
    VScroll1.Max = Abs(picMapIn.Height - picMapOut.Height) / lScrollingValue
    gbFmSWSetupLoaded = True

End Sub

Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
    If UnloadMode = vbManual Then

```

```

    ModalResult = False
End If
End Sub

```

```

Private Sub Form_Unload(Cancel As Integer)
    Unload fmSelectCats
    gbFmSWSetupLoaded = False
End Sub

```

```

Private Sub SetupForm()

    On Error GoTo ErrorHandler

    Dim Count As Long, lNum As Long, lLine As Long
    Dim sngTop As Single
    Dim nIndex As Integer
    Dim li As ListItem, li2 As ListItem

    sngTop = txtMap(0).Top
    For Count = 1 To glCustomCategoriesNum
        Load txtMap(Count)
        txtMap(Count).Top = sngTop
        txtMap(Count).Text = ""
        txtMap(Count).Visible = True

        Load lblNum(Count)
        lblNum(Count).Top = txtMap(Count).Top + (txtMap(Count).Height -
lblNum(Count).Height) / 2
        lblNum(Count).Caption = CStr(Count)
        lblNum(Count).Visible = True

        sngTop = txtMap(Count).Top + txtMap(Count).Height + 10
    Next Count
    picMapIn.Height = sngTop + 30 '- txtMap(0).Height

    'Set up info
    txtName.Text = SWPackageCurrent.Name
    txtSWDir.Text = SWPackageCurrent.VulnDB.DBPathName

    If txtSWDir.Text <> "" Then
        'Load tables from DB
        Call cmdLoadTables_Click

        'Set up table
        nIndex = -1
        For Count = 0 To (cbTables.ListCount - 1)
            If cbTables.List(Count) = SWPackageCurrent.VulnDB.TableName Then
                nIndex = Count
                Exit For
            End If
        Next Count
        cbTables.ListIndex = nIndex

        'Set up ID Field
        nIndex = -1
        For Count = 0 To (cbID.ListCount - 1)
            If cbID.List(Count) = SWPackageCurrent.VulnDB.FieldName_ID Then
                nIndex = Count
                Exit For
            End If
        Next Count
        cbID.ListIndex = nIndex

        'Set up Description Field
        nIndex = -1
        For Count = 0 To (cbDescription.ListCount - 1)
            If cbDescription.List(Count) = SWPackageCurrent.VulnDB.FieldName_Description
Then
                nIndex = Count
                Exit For
            End If
        Next Count
        cbDescription.ListIndex = nIndex

        'Load categories
        Call cmdLoadCatInfo_Click
    
```

```

'Main categories
For Count = 1 To UBound(SWPackageCurrent.VulnDB.MainCategories)
    Set li2 =
lvSWCategories.FindItem(CStr(SWPackageCurrent.VulnDB.MainCategories(Count).ID),
lvwText)
    Set li = lvMainCats.ListItems.Add(, "Key" &
SWPackageCurrent.VulnDB.MainCategories(Count).ID,
SWPackageCurrent.VulnDB.MainCategories(Count).Number)
        li.SubItems(1) = SWPackageCurrent.VulnDB.MainCategories(Count).ID
        li.SubItems(2) = li2.SubItems(1)
        li.SubItems(3) = SWPackageCurrent.VulnDB.MainCategories(Count).Sequence
    Next Count

'Set up mapping info
lNum = txtMap.Count - 1
For Count = 1 To lNum
    txtMap(Count).Text = SWPackageCurrent.VulnDB.Mapping(Count)
Next Count

Else
    cbTables.Clear
    cbTables.Enabled = False
    lblTables.Enabled = False
    cmdLoadTables.Enabled = False
    cmdLoadCatInfo.Enabled = False
    frmMappingInfo.Visible = False
End If

Quit:
Exit Sub

ErrorHandler:
If lLine = 1 Then
    lNum = 0
    Resume Next
End If
Resume Quit

End Sub

```

```

Private Sub frmMappingInfo_DblClick()

    Dim lTemp As Long

    If lButton = (vbCtrlMask + vbShiftMask + vbAltMask) Then
        If frmMain(0).Visible Then
            lvMainCats.ListItems.Clear

            Call AddListItem(lvMainCats, "Key1000", "1", "1000", "Information Gathering and Recon",
"1001,1002,1003,1004,1005,1006,1007,1008,1009,1010,1011,1012,1013,1014,1016,1017,1018,
1019,1021,1023,1024,1026,1028,1032,1033,1034,1035,1036,1037,1038,1039,1040,1041")
            Call AddListItem(lvMainCats, "Key2000", "2", "2000", "File Transfer Protocols",
"2001,2002,2003,2004,2005,2006,2007,2010,2011,2012,2013,2014,2016,2017,2018,2019,2021,
2024")
            Call AddListItem(lvMainCats, "Key3000", "3", "3000", "Hardware Peripherals",
"3001,3002,3003,3006,3007,3008,3009,3010,3011,3012")
            Call AddListItem(lvMainCats, "Key4000", "4", "4000", "Backdoors and Misconfigurations",
"4001,4002,4004,4005,4006,4007,4008,4009,4010")
            Call AddListItem(lvMainCats, "Key5000", "5", "5000", "SMTP and Mail Transfer",
"5001,5002,5003,5005,5006,5007,5008,5009,5011,5013,5014,5015,5016,5017,5018,5019,5020,
5021,5023")
            Call AddListItem(lvMainCats, "Key6000", "6", "6000", "Remote Procedure Call Services",
"6003,6004,6005,6007,6008,6009,6014,6015,6016,6019,6020,6021,6025,6027,6028,6034,6035,
6036,6037")
            Call AddListItem(lvMainCats, "Key7000", "7", "7000", "Networked File Systems",
"7001,7002,7003,7004,7005,7006,7007,7008,7010,7011,7013,7014")
            Call AddListItem(lvMainCats, "Key8000", "8", "8000", "Denial of Service Attacks",
"8001,8002,8003,8004,8005,8006,8007,8008,8009,8010,8011,8012,8016,8017,8019,8020,8023,
8024,8025,8026,8027,8028,8029,8030,8031,8032,8033,8034,8035,8036,8038,8039,8040,8041,8
042,8043,8044,8046,8049,8050,8051,8053,8054")
            Call AddListItem(lvMainCats, "Key9000", "9", "9000", "Password Guessing/Grinding",
"9001,9002,9003,9004,9005,9006,9007")
            Call AddListItem(lvMainCats, "Key10000", "10", "10000", "World Wide Web, HTTP, and CGI",

```



```

"10001,10002,10003,10004,10006,10008,10009,10010,10012,10014,10015,10016,10017,10018,1
0019,10020,10021,10022,10023,10024,10025,10026,10027,10028,10029,10030,10031,10032,100
33,10034,10035,10036,10037,10038,10039,10040,10042,10043,10044,10046,10047,10048,10049
,10050,10053,10054,10055,10056,10057,10064,10065,10066,10067,10068")
    Call AddListItem(lvMainCats, "Key11000", "11", "11000", "Network Protocol
Spoofing", "11006,11010,11011")
    Call AddListItem(lvMainCats, "Key12000", "12", "12000", "Packet Filter
Verification Tests",
"12001,12002,12003,12004,12005,12006,12007,12008,12009,12010,12011,12012,12013,12014,1
2015,12016,12017,12018,12019,12020,12021,12022,12023,12024,12025,12026,12027,12028,120
29,12030,12031,12032,12033,12034,12035,12036,12040,12041,12042,12043,12044,12045,12046
,12047,12048,12049,12050,12051,12052,12060,12061,12062,12070,12071,12072")
    Call AddListItem(lvMainCats, "Key13000", "13", "13000", "Firewalls, Filters, and
Proxies", "13001,13002,13005,13011,13012,13013")
    Call AddListItem(lvMainCats, "Key14000", "14", "14000", "Authentication
Mechanisms", "14001,14002,14003,14004,14005,14006,14007")
    Call AddListItem(lvMainCats, "Key15000", "15", "15000", "General Remote
Services",
"15001,15003,15004,15005,15006,15007,15008,15009,15011,15014,15015,15020,15021,15024,1
5025,15026,15027,15028,15029,15030,15031,15032,15033,15034,15035,15036,15037,15038,150
39,15040,15043,15044,15045,15047,15048")
    Call AddListItem(lvMainCats, "Key16000", "16", "16000", "SMB/NetBIOS Resource
Sharing",
"16001,16002,16003,16004,16005,16006,16007,16008,16009,16020,16021,16022,16023,16024")
    Call AddListItem(lvMainCats, "Key17000", "17", "17000", "Domain Name System and
BIND",
"17002,17004,17005,17007,17008,17010,17014,17018,17020,17021,17022,17023,17024")
    Call AddListItem(lvMainCats, "Key18000", "18", "18000", "Windows NT - Network
Vulnerabilities",
"18001,18002,18003,18004,18005,18007,18008,18009,18010,18011,18012,18013,18014,18015,1
8016,18017,18018,18019,18020,18021,18022,18023,18024,18025,18026,18027,18028,18029,180
30,18031")
    Call AddListItem(lvMainCats, "Key20000", "20", "20000", "SNMP/Network
Management",
"20001,20010,20011,20012,20013,20014,20015,20016,20020,20022,20023,20024,20030")
    Call AddListItem(lvMainCats, "Key21000", "21", "21000", "Network Port Scanning",
"21001,21002,21003,21004,21005,21006,21007")
    Call AddListItem(lvMainCats, "Key22000", "22", "22000", "Windows NT - Browser
Zone Policy",
"22001,22002,22003,22004,22005,22006,22007,22008,22009,22010,22011,22012,22013,22014,2
2015,22016,22017,22018,22019,22020,22021,22022")
    Call AddListItem(lvMainCats, "Key23000", "23", "23000", "Windows NT - Privilege
Enumeration",
"23001,23002,23003,23004,23005,23006,23007,23008,23009,23010,23011,23012,23013,23014,2
3015,23016,23017,23018,23019,23020,23021,23022,23023,23024,23025,23026,23027,23028,230
29,23030,23031")
    Call AddListItem(lvMainCats, "Key24000", "24", "24000", "Windows NT - Local
System Policy",
"24001,24002,24003,24004,24005,24006,24007,24008,24009,24010,24011,24012,24013,24014,2
4015,24016,24017,24018,24019,24020,24022,24023,24024,24025,24026,24027")
    Call AddListItem(lvMainCats, "Key25000", "25", "25000", "Windows NT - Auditing
and Password Policy",
"25001,25002,25003,25004,25005,25006,25007,25008,25009,25010,25011,25012,25013,25014,2
5015,25016,25017,25018,25019,25020,25021,25022,25023")
    Call AddListItem(lvMainCats, "Key26000", "26", "26000", "Windows NT -
Information Gathering", "26001,26002,26003,26004,26005,26006,26007,26008,26009,26010")
    Call AddListItem(lvMainCats, "Key27000", "27", "27000", "Intrusion Detection
System Verification",
"27001,27002,27003,27004,27005,27006,27007,27008,27009,27010,27011,27012,27013,27014,2
7015,27016,27017,27018,27019,27020,27021,27022,27023,27024,27025")
    Call AddListItem(lvMainCats, "Key28000", "28", "28000", "Windows NT - Service
Packs (SP) and Hot Fixes (HF)",
"28001,28002,28005,28006,28010,28011,28012,28013,28014,28015,28016,28017,28018,28019,2
8020,28021,28022,28023,28024,28025,28026,28027,28028,28029,28030,28031,28032,28033,280
34,28035,28036,28037,28038,28039,28040,28041,28042,28043,28080,28081,28082,28083,28084
,28085,28086,28087,28088,28089,28090,28091,28092,28150,28151,28152,28153,28154,28155,2
8156,28157,28158,28159,28160,28161,28162,28163,28164,28165,28166,28167,28168,28169,281
73,28174,28175,28176,28200,28201,28250,28251,28252,28253,28254,28255,28256")
    Call AddListItem(lvMainCats, "Key29000", "29", "29000", "Windows NT - Third
Party Software",
"29001,29002,29003,29008,29009,29010,29011,29012,29013,29014,29015,29016,29017,29018,2
9019,29021,29022,29023")
    Call AddListItem(lvMainCats, "Key30000", "30", "30000", "Windows NT - Services
", "30001,30002,30003,30004,30005,30006,30007,30008,30009,30010,30012")
    Call AddListItem(lvMainCats, "Key31000", "31", "31000", "Windows NT - Remote
Access Server",
"31001,31002,31003,31004,31005,31006,31007,31008,31009,31010,31011,31012")

```

```

Else
    Call AddText(1, "1001,1002,1004,1038,1039,2006,2018,2019,2024,3001-
3003,3006,3008-3011,9000-
9007,10032,10033,15005,15007,15025,15040,15043,16001,16002,16024,17020,18002,18004,180
05,18007-18009,18015,18021,31006")
    Call AddText(2, "1000,1006-
1008,1010,1016,1017,1019,1024,1032,1033,1036,1039,1040,1041,2006,2010,3008,3009,5003,5
006,5007,5011,6009,7005,7014,10002,10010,10015,10020,10021,10038,10047,10048,10064,120
01-12010,12012-12036,13012,14004-14006,15007,15028,16003-
16005,16020,17000,17002,17004,17005,17007,17008,17010,17014,17018,17020-
17023,18004,18005,18007,18026,20001,20010-20016,20020,20022,20023,20030,21000-
21002,21006,21007,26000,26004-26006,26008-26010")
    Call AddText(3, "1001-
1005,1023,1035,3008,3009,5005,6005,15025,15032,15035,20024,26001-26003,26007")
    Call AddText(4, "2001,3008,3009,4000-4002,4004-4010,5001,15003,18015")
    Call AddText(5, "1005,1009-1014,1018,1021,1026,1028,1034,1037,2000,2003-
2007,2012,2016,2017,2021,3006-3009,3012,5000,5002,5008,5009,5015,5019,6000,6003-
6005,6007-6009,6014-6016,6019-6021,6025,6027,6028,6034-6037,7000,7004-
7008,7010,7011,7013,7014,10000,10002-10004,10006,10008,10009,10012,10014-10021,10027-
10030,10034-10036,10039,10040,10043,10044,10046,10049,10050,10053-10056,10064-
10068,13005,13011,13013,14000-14003,15000,15003-
15006,15008,15009,15011,15014,15015,15021,15024,15026-15040,15043-
15045,15047,15048,16000-16009,16020-16024,17000,17007,17024,18000-
18002,18011,18012,18024,18025,20000,20001,21000-21007,26004-26006,26008-26010,30000-
30010,30012,31001-31004,31007-31012")
    Call AddText(6, "1009,2010-
2014,2021,5002,5008,5009,5013,5017,5018,6003,6004,6016,6037,7002,7006-
7008,10031,10034,10037,10066,10067,13013,15004,15006,15015,15024,15027,15036,15039,180
03,18010,18013,18014,18017-18020,18031,23000-23031")
    Call AddText(7, "2007,5016,5021,6016,8001,11000,11006,11010,11011,13000-
13002,15045,17020-17023")
    Call AddText(8, "2002,2017,3003,3010,3011,4000,5015,5021,6003,6009,7001-
7003,7006,8001,10008,10025,12000-12009,12011-12036,12040-12052,12060-12062,12070-
12072,13005,13012,13013,14001-14003,14005,15001,15008,15009,15021,15043,15045,16006-
16008,16020-16023,17004,18000,18001,18007-18015,18017-18020,18022-18031,20001,20010-
20016,20020,20022-20024,20030,22000-22022,23000-23031,24000-24020,24022,24025-
24027,25000-25023,26004,27002,29008-29019,29021-29023,30001,30002,30005,31004,31006")
    Call AddText(9,
"2005,2018,2019,3007,3012,5009,5011,5013,5014,5018,5020,5023,6015,6027,8000-
8012,8016,8017,8019,8020,8023-8036,8038-8044,8046,8049-
8051,8053,8054,10001,10014,10022-
10024,10026,10035,10042,10057,13011,14007,15015,15027,15029,15034-
15036,15039,15040,15047,15048,16024,17024,18016,20001")
    Call AddText(10, "")
    Call AddText(11, "3000-3003,3006-3012,13001")
    Call AddText(12, "27000-27025,28000-28002,28005,28006,28010-28043,28080-
28092,28150-28169,28173-28176,28200,28201,28250-28256,29000-29003,29008-29019,29021-
29023")
    Call AddText(13, "24000-24020,24022,24025,24026,24027,25000-25023,31001-
31003,31005,31007-31010,31012")
    Call AddText(14, "")
    Call AddText(15, "5006,5007,5020,15038,21001-21005")
End If
End If
End Sub

Private Sub AddText(lTextBoxNum As Long, sString As String)
    If lTextBoxNum < txtMap.Count Then
        txtMap(lTextBoxNum).Text = sString
    End If
End Sub

Private Sub AddListItem(lv As ListView, sKey As String, sNumber As String, sID As
String, sDescription As String, sSeq As String)

    Dim li As ListItem

    Set li = lv.ListItems.Add(, sKey, sNumber)
    li.SubItems(1) = sID
    li.SubItems(2) = sDescription
    li.SubItems(3) = sSeq

End Sub

Private Sub lblNum_Click(Index As Integer)
    txtMap(Index).SetFocus

```

```

End Sub
-----
Private Sub lvMainCats_DblClick()
    Call cmdCatsIn_Click
End Sub
-----
Private Sub lvSWCategories_DblClick()
    Call cmdMap_Click
End Sub
-----
Private Sub TabStrip1_Click()
    Call SetTab(TabStrip1, frmMain)
    cmdRemove.Visible = frmMain(0).Visible
End Sub
-----
Private Sub txtMap_GotFocus(Index As Integer)
    Call SetAllLblNumFontBold
    lblNum(Index).FontBold = True
    lblMapping.Caption = "Mapping to custom category " & Index
    lblMapping.Visible = True
End Sub
-----
Private Sub txtMap_LostFocus(Index As Integer)

    Dim bOK As Boolean
    Dim Count As Long

    bOK = False
    For Count = 1 To lblNum.Count - 1
        If lblNum(Count).FontBold Then
            bOK = True
            Exit For
        End If
    Next Count
    lblMapping.Visible = bOK

End Sub
-----
Private Sub txtSWSDir_Change()

    Dim sTemp As String, sDBName As String
    Dim Place As Long

    sTemp = txtSWSDir.Text

    Place = InStrRev(sTemp, "\")
    If Place > 0 Then
        sDBName = Mid$(sTemp, Place + 1)
    Else
        sDBName = sTemp
    End If

    If Dir(sTemp, vbArchive + vbHidden + vbNormal + vbReadOnly + vbSystem + vbVolume) =
sDBName Then
        cbTables.Enabled = True
        cmdLoadTables.Enabled = True
        lblTables.Enabled = True
    Else
        cbTables.Enabled = False
        cmdLoadTables.Enabled = False
        cmdLoadCatInfo.Enabled = False
        lblTables.Enabled = False
    End If

End Sub
-----
Private Sub LoadTableFields()

    On Error GoTo ErrorHandler

    Dim TempConn As New ADODB.Connection
    Dim TempRS As New ADODB.Recordset
    Dim Count As Long
    Dim li As ListItem

    If cbTables.Text = "" Then Exit Sub

```

```

'Open connection
TempConn.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" &
txtSWDir.Text & ";Persist Security Info=False"
TempConn.Open

'Open table
TempRS.CursorLocation = adUseServer
TempRS.CursorType = adOpenStatic
TempRS.LockType = adLockOptimistic
TempRS.Open "SELECT TOP 2 * FROM " & cbTables.Text, TempConn, adCmdTable

cbID.Clear
lvExample.ColumnHeaders.Clear
lvExample.ListItems.Clear
cbDescription.Clear
If Not TempRS.EOF Then
    TempRS.MoveFirst
    TempRS.MoveNext
End If
For Count = 0 To (TempRS.Fields.Count - 1)
    cbID.AddItem TempRS.Fields(Count).Name
    cbDescription.AddItem TempRS.Fields(Count).Name

'Setup example headers
Call lvExample.ColumnHeaders.Add(, "Key" & Count, TempRS.Fields(Count).Name) ',
frmMain.TextWidth(TempRS.Fields(Count).Name) + 50)

'Setup example data
If Not TempRS.EOF Then
    If Count = 0 Then
        Set li = lvExample.ListItems.Add(, "KeyLI",
TempRS.Fields(TempRS.Fields(Count).Name).Value)
    Else
        li.SubItems(Count) = TempRS.Fields(TempRS.Fields(Count).Name).Value
    End If
End If
Next Count
If cbID.ListCount > 0 Then
    cbID.ListIndex = 0
    cbDescription.ListIndex = 0
End If

Quit:
Exit Sub

ErrorHandler:
Resume Quit

End Sub

Private Sub VScroll1_Change()
'move picture up/down
picMapIn.Top = -(VScroll1.Value * lScrollingValue)
End Sub

Private Sub VScroll1_Scroll()
'move picture up/down
picMapIn.Top = -(VScroll1.Value * lScrollingValue)
End Sub

Private Sub SetAllLblNumFontBold(Optional bValue As Boolean = False)

Dim Count As Long

For Count = 1 To (lblNum.Count - 1)
    lblNum(Count).FontBold = bValue
Next Count
lblMapping.Visible = False

End Sub

Private Function CheckInfo() As Boolean

Dim bOK As Boolean, bAllEmpty As Boolean
Dim Count As Long

bOK = True

```

```

'Check name
If (txtName.Text = "") Or (LCase(txtName.Text) = "<new package>") Then
    bOK = False
    Call MsgBox("Please specify a valid name!", vbOKOnly + vbInformation)
    txtName.SetFocus
    GoTo Quit
End If

'Check db path
If (txtSwdDir.Text = "") Then
    bOK = False
    Call MsgBox("Please specify a valid database!", vbOKOnly + vbInformation)
    txtSwdDir.SetFocus
    GoTo Quit
End If

'Check tables
If (cbTables.Text = "") Then
    bOK = False
    Call MsgBox("Please specify a valid table from the database!", vbOKOnly +
vbInformation)
    cbTables.SetFocus
    GoTo Quit
End If

'Check ID Field
If (cbID.Text = "") Then
    bOK = False
    Call MsgBox("Please specify a valid field representing the unique ID for the
categories!", vbOKOnly + vbInformation)
    cbID.SetFocus
    GoTo Quit
End If

'Check tables
If (cbDescription.Text = "") Then
    bOK = False
    Call MsgBox("Please specify a valid field representing the description for the
categories!", vbOKOnly + vbInformation)
    cbDescription.SetFocus
    GoTo Quit
End If

'Check main categories
If lvMainCats.ListItems.Count < 1 Then
    bOK = False
    Call MsgBox("You have to set up the main software categories, as well as the
categories belonging to them!", vbOKOnly + vbInformation)
    TabStrip1.Tabs(1).Selected = True
    lvMainCats.SetFocus
    GoTo Quit
End If

'Check mapping
bAllEmpty = True
For Count = 1 To (txtMap.Count - 1)
    If txtMap(Count).Text <> "" Then
        bAllEmpty = False
        Exit For
    End If
Next Count
If bAllEmpty Then
    bOK = False
    Call MsgBox("You have to map the software categories to the custom categories!",
vbOKOnly + vbInformation)
    cmdMap.SetFocus
    GoTo Quit
End If

'Set up info
With SWPackageCurrent
    .Name = txtName.Text

    .VulnDB.DBPathName = txtSwdDir.Text
    .VulnDB.TableName = cbTables.Text
    .VulnDB.FieldName_ID = cbID.Text

```

```

.VulnDB.FieldName_Description = cbDescription.Text

'Main categories
ReDim .VulnDB.MainCategories(1 To lvMainCats.ListItems.Count)
For Count = 1 To lvMainCats.ListItems.Count
    .VulnDB.MainCategories(Count).Number = CLng(lvMainCats.ListItems(Count).Text)
    .VulnDB.MainCategories(Count).ID = CLng(lvMainCats.ListItems(Count).SubItems(1))
    .VulnDB.MainCategories(Count).Description =
lvMainCats.ListItems(Count).SubItems(2)
    .VulnDB.MainCategories(Count).Sequence = lvMainCats.ListItems(Count).SubItems(3)
Next Count

'Mapped categories
ReDim .VulnDB.Mapping(1 To glCustomCategoriesNum)
For Count = 1 To glCustomCategoriesNum
    .VulnDB.Mapping(Count) = txtMap(Count).Text
Next Count

End With

Quit:
    CheckInfo = bOK

End Function

```

```

Public Sub ReorderMainCategories()

    On Error GoTo ErrorHandler

    Dim Count As Long, lNumItems As Long
    Dim TempArray() As CustomCategoryInfoType
    Dim TempItem As CustomCategoryInfoType
    Dim li As ListItem
    Dim lCurIndex As Long
    Dim sCurID As String

    lNumItems = lvMainCats.ListItems.Count
    If lNumItems < 1 Then Exit Sub

    ReDim TempArray(1 To lNumItems)
    For Count = 1 To lNumItems
        TempItem.Number = CLng(lvMainCats.ListItems(Count).Text)
        TempItem.ID = CLng(lvMainCats.ListItems(Count).SubItems(1))
        TempItem.Description = lvMainCats.ListItems(Count).SubItems(2)
        TempItem.Sequence = lvMainCats.ListItems(Count).SubItems(3)
        TempItem.Key = lvMainCats.ListItems(Count).Key
        Call AddItemToOrderedArray(TempArray, TempItem)
    Next Count

    lCurIndex = 0
    sCurID = lvMainCats.SelectedItem.SubItems(1)
    lvMainCats.ListItems.Clear
    For Count = 1 To lNumItems
        Set li = lvMainCats.ListItems.Add(, TempArray(Count).Key, TempArray(Count).Number)
        li.SubItems(1) = TempArray(Count).ID
        li.SubItems(2) = TempArray(Count).Description
        li.SubItems(3) = TempArray(Count).Sequence

        If sCurID = li.SubItems(1) Then
            lCurIndex = Count
        End If
    Next Count
    If (lCurIndex = 0) And (lvMainCats.ListItems.Count > 0) Then lCurIndex = 1
    If lCurIndex > 0 Then
        lvMainCats.ListItems(lCurIndex).Selected = True
        Call lvMainCats.ListItems(lCurIndex).EnsureVisible
    End If

    Exit Sub

ErrorHandler:
    lCurIndex = 0
    If lvMainCats.ListItems.Count > 0 Then lCurIndex = 1

End Sub

```

```

Private Sub AddItemToOrderedArray(OrderedArray() As CustomCategoryInfoType, AddItem As
CustomCategoryInfoType)

    Dim Count As Long, lIndex As Long
    Dim bFound As Boolean

    lIndex = 1
    For Count = 1 To UBound(OrderedArray)
        If (OrderedArray(Count).ID = 0) Then
            lIndex = Count
            Exit For
        Else
            If AddItem.ID < OrderedArray(Count).ID Then
                lIndex = Count
                Exit For
            End If
        End If
    Next Count

    If (lIndex < UBound(OrderedArray)) Then
        If OrderedArray(lIndex).ID <> 0 Then
            For Count = UBound(OrderedArray) To lIndex + 1 Step -1
                OrderedArray(Count) = OrderedArray(Count - 1)
            Next Count
        End If
    End If

    OrderedArray(lIndex) = AddItem

End Sub

```

```

Public Sub GetStuff()

    Dim Count As Long
    Dim sTemp As String

    sTemp = ""
    For Count = 1 To lvMainCats.ListItems.Count
        sTemp = sTemp & "Call AddListItem(lvMainCats, " & Chr(34) & "Key" &
lvMainCats.ListItems(Count).SubItems(1) & Chr(34) & ", "
        sTemp = sTemp & Chr(34) & lvMainCats.ListItems(Count).Text & Chr(34) & ", "
        sTemp = sTemp & Chr(34) & lvMainCats.ListItems(Count).SubItems(1) & Chr(34) & ", "
        sTemp = sTemp & Chr(34) & lvMainCats.ListItems(Count).SubItems(2) & Chr(34) & ", "
        sTemp = sTemp & Chr(34) & lvMainCats.ListItems(Count).SubItems(3) & Chr(34) &
vbCrLf
    Next Count

    Debug.Print sTemp
End Sub

```

B.2 SOURCE CODE FOR THE “MODMAIN” MODULE

The “modMain” module is used to hold code for global variables as well as global functions and procedures .

The source code for this module follows below.

```

Option Explicit

Global lLastBlah As Long
Global gsConnectionStringToMainDB As String
Global SQL As String
Global gbDBDetected As Boolean

'Options
Global gsPreviousPaths() As String

```

```

Global gsPreviousComparePaths() As String
Global gsAdjectives() As String

'Variables
Global BrowseInfo As BrowseInfo
Global InitDir As String
Global gsFilter As String
Global gsStatusTextFound As String
Global glNumDatabaseScans As Long
Global gsDefaultConnString As String
Global gnCurCat As Integer
Global ModalResult As Boolean, gbBusyAddEdit As Boolean

Global glNumVulnerabilityCategories As Long
Global glNumCustomVulnerabilityCategories As Long

'Constants
Global Const xPLUS As Long = 0
Global Const xMINUS As Long = 1
Global Const xMULTIPLY As Long = 2
Global Const xDEVIDE As Long = 3
Global Const xINFINITY As Long = 4

Global Const xEQUALS As Long = 0
Global Const xLESS_THAN As Long = 1
Global Const xGREATER_THAN As Long = 2
Global Const xLESS_THAN_OR_EQUAL As Long = 3
Global Const xGREATER_THAN_OR_EQUAL As Long = 4

'Form variables
Global gbFmCalculationsLoaded As Boolean
Global gbFmGraphicsLoaded As Boolean
Global gbFmHelpLoaded As Boolean
Global gbFmOptionsLoaded As Boolean
Global gbFmSetupLoaded As Boolean
Global gbFmSetupNames As Boolean
Global gbFmSWSetupLoaded As Boolean

'Windows functions
Public Declare Function SHBrowseForFolder Lib "shell32" (lpbi As BrowseInfo) As Long
Public Declare Function SHGetPathFromIDLList Lib "shell32" (ByVal pidList As Long,
ByVal lpBuffer As String) As Long
Public Declare Function lstrcat Lib "kernel32" Alias "lstrcatA" (ByVal lpString1 As
String, ByVal lpString2 As String) As Long
Public Declare Function PostMessage Lib "user32" Alias "PostMessageA" (ByVal hwnd As
Long, ByVal wParam As Long, ByVal lParam As Long) As Long
Public Declare Function SendMessage Lib "user32" Alias "SendMessageA" (ByVal hwnd As
Long, ByVal wParam As Long, ByVal lParam As Any) As Long
Public Declare Function SendMessageStr Lib "user32" Alias "SendMessageA" (ByVal hwnd
As Long, ByVal wParam As Long, ByVal lParam As Long, ByVal lParam As Any) As Long

'Types
Public Type BrowseInfo
    hwndOwner As Long
    pidlRoot As Long
    pszDisplayName As Long
    lpszTitle As Long
    ulFlags As Long
    lpfnCallback As Long
    lParam As Long
    iImage As Long
End Type

Public Type GroupingType
    ScanFrom As Long
    ScanTo As Long
    Adjective As String

    VulnerabilityFrom As Long
    VulnerabilityTo As Long
    VulnerabilityTranslatedFrom As Double
    VulnerabilityTranslatedTo As Double

    Cx_From As Double
    Cx_To As Double

    Mu_Lowerbound As Double

```



```

    Mu_Upperbound As Double

    MIN_Cx_Mu_LB As Double
    MIN_Cx_Mu_UB As Double
End Type

Public Type AdjectiveType
    LowerOperator As Long
    LowerValue As String
    UpperOperator As Long
    UpperValue As String
End Type

Public Type ScanInfoType
    TableName As String
    VulnCount() As Long
    VulnID() As Long
    MaxEntries As Long
End Type
Global ScanInfo() As ScanInfoType
Global ActualScan As ScanInfoType

Public Type PredictionType
    Lowerbound As Long
    Upperbound As Long
End Type
Public Type PredictionDblType
    Lowerbound As Double
    Upperbound As Double
End Type

Public Type CategoryType
    CategoryNumber As Long
    NumberOfVulnerabilities() As Long
    MaxVulnerabilityValue As Long
    NumberOfGroups As Long
    Groups() As GroupingType
    Rule_Name As String
    Rule_Value As String

    Adjectives() As AdjectiveType
    HasBeenSetup As Boolean

    Membership_Op1 As Long
    Membership_Devider As Long
    Membership_Op2 As Long
    Membership_Op3 As Long
    Membership_To As Long
    Membership_Op4 As Long

    MAXofMIN_Cx_Mu As PredictionDblType

    Final As PredictionDblType
    Final_AmtVulns As PredictionType

    DisplayResultOnGraph As Boolean
End Type
Global Categories() As CategoryType

Public Type CustomCategoryInfoType
    ID As Long
    Number As Long
    Description As String

    Key As String
    Sequence As String
End Type
Global CustomCategoryInfo() As CustomCategoryInfoType
Global glCustomCategoriesNum As Long

Public Type OptionsType
    Prediction_LineColor As OLE_COLOR
    Prediction_ColumnColor As OLE_COLOR
    SaveLoad As Integer
End Type
Global Options As OptionsType

```



```

Public Type ScanningSWInfoType
    Name As String
    MappingInfo() As String
End Type

Public Type SWVulnsDescriptionIDType
    DBPathName As String
    TableName As String
    FieldName_ID As String
    FieldName_Description As String

    MainCategories() As CustomCategoryInfoType
    Mapping() As String
End Type

Public Type ScanningDBSampleType
    DBPathName As String
    CreatedFrom As Long
    SQLResultTableName As String
End Type

Public Type ScanningDBType
    TableName As String
    FieldName As String
    SampleDB As ScanningDBSampleType
End Type

Public Type SWPackageInfoType
    Number As Long
    Name As String
    VulnDB As SWVulnsDescriptionIDType
    ScanningDB As ScanningDBType
End Type

Global SWPackageInfo() As SWPackageInfoType
Global SWPackageCurrent As SWPackageInfoType

Global Const MAX_PATH = 260

'ulFlags Constants
Global Const BIF_BROWSEFORCOMPUTER = &H1000        '0x1000 // Browsing for Computers.
Global Const BIF_BROWSEFORPRINTER = &H2000        '0x2000 // Browsing for Printers
Global Const BIF_BROWSEINCLUDEFILES = &H4000        '0x4000 // Browsing for Everything

Global Const BIF_RETURNONLYFSDIRS = &H1            '0x0001 // For finding a folder to
start document searching
Global Const BIF_DONTGOBELOWDOMAIN = &H2          '0x0002 // For starting the Find
Computer
Global Const BIF_STATUSTEXT = &H4                  '0x0004
Global Const BIF_RETURNFSANCESTORS = &H8          '0x0008
Global Const BIF_EDITBOX = &H10                   '0x0010
Global Const BIF_VALIDATE = &H20                  '0x0020 // insist on valid result
(or CANCEL)

'pIDLRoot Constants
Global Const CSIDL_DESKTOP = &H0                   '0x0000
Global Const CSIDL_INTERNET = &H1                  '0x0001
Global Const CSIDL_PROGRAMS = &H2                  '0x0002
Global Const CSIDL_CONTROLS = &H3                  '0x0003
Global Const CSIDL_PRINTERS = &H4                  '0x0004
Global Const CSIDL_PERSONAL = &H5                  '0x0005
Global Const CSIDL_FAVORITES = &H6                 '0x0006
Global Const CSIDL_STARTUP = &H7                   '0x0007
Global Const CSIDL_RECENT = &H8                    '0x0008
Global Const CSIDL_SENDTO = &H9                    '0x0009
Global Const CSIDL_BITBUCKET = &HA                  '0x000a
Global Const CSIDL_STARTMENU = &HB                 '0x000b
Global Const CSIDL_DESKTOPDIRECTORY = &H10         '0x0010
Global Const CSIDL_DRIVES = &H11                   '0x0011
Global Const CSIDL_NETWORK = &H12                  '0x0012
Global Const CSIDL_NETHOOD = &H13                  '0x0013
Global Const CSIDL_FONTS = &H14                    '0x0014
Global Const CSIDL_TEMPLATES = &H15                '0x0015
Global Const CSIDL_COMMON_STARTMENU = &H16         '0x0016
Global Const CSIDL_COMMON_PROGRAMS = &H17         '0x0017
Global Const CSIDL_COMMON_STARTUP = &H18          '0x0018
Global Const CSIDL_COMMON_DESKTOPDIRECTORY = &H19 '0x0019
Global Const CSIDL_APPDATA = &H1A                  '0x001a

```

```

Global Const CSIDL_PRINTHOOD = &H1B           '0x001b
Global Const CSIDL_ALTSTARTUP = &H1D         '0x001d
Global Const CSIDL_COMMON_ALTSTARTUP = &H1E  '0x001e
Global Const CSIDL_COMMON_FAVORITES = &H1F   '0x001f
Global Const CSIDL_INTERNET_CACHE = &H20     '0x0020
Global Const CSIDL_COOKIES = &H21            '0x0021
Global Const CSIDL_HISTORY = &H22            '0x0022

'GetCallbackProcAddress Constants
Public Const WM_USER = &H400

'GetCallbackProcAddress Constants (Messages)
Global Const BFFM_SETSTATUSTEXTA = (WM_USER + 100) '(WM_USER + 100)
Global Const BFFM_ENABLEOK = (WM_USER + 101)      '(WM_USER + 101)
Global Const BFFM_SETSELECTIONA = (WM_USER + 102) '(WM_USER + 102)
Global Const BFFM_SETSELECTIONW = (WM_USER + 103) '(WM_USER + 103)
Global Const BFFM_SETSTATUSTEXTW = (WM_USER + 104) '(WM_USER + 104)

'GetCallbackProcAddress Constants (uMsg)
Global Const BFFM_INITIALIZED = 1               '1
Global Const BFFM_SELCHANGED = 2               '2
Global Const BFFM_VALIDATEFAILED_A = 3         '3
Global Const BFFM_VALIDATEFAILED_W = 4         '4

Sub Main()

'1: 1001,1002,1004,1038,1039,2006,2018,2019,2024,3001-3003,3006,3008-3011,9000-
9007,10032,10033,15005,15007,15025,15040,15043,16001,16002,16024,17020,18002,18004,180
05,18007-18009,18015,18021,31006
'2: 1000,1006-
1008,1010,1016,1017,1019,1024,1032,1033,1036,1039,1040,1041,2006,2010,3008,3009,5003,5
006,5007,5011,6009,7005,7014,10002,10010,10015,10020,10021,10038,10047,10048,10064,120
01-12010,12012-12036,13012,14004-14006,15007,15028,16003-
16005,16020,17000,17002,17004,17005,17007,17008,17010,17014,17018,17020-
17023,18004,18005,18007,18026,20001,20010-20016,20020,20022,20023,20030,21000-
21002,21006,21007,26000,26004-26006,26008-26010
'3: 1001-1005,1023,1035,3008,3009,5005,6005,15025,15032,15035,20024,26001-26003,26007
'4: 2001,3008,3009,4000-4002,4004-4010,5001,15003,18015
'5: 1005,1009-1014,1018,1021,1026,1028,1034,1037,2000,2003-
2007,2012,2016,2017,2021,3006-3009,3012,5000,5002,5008,5009,5015,5019,6000,6003-
6005,6007-6009,6014-6016,6019-6021,6025,6027,6028,6034-6037,7000,7004-
7008,7010,7011,7013,7014,10000,10002-10004,10006,10008,10009,10012,10014-10021,10027-
10030,10034-10036,10039,10040,10043,10044,10046,10049,10050,10053-10056,10064-
10068,13005,13011,13013,14000-14003,15000,15003-
15006,15008,15009,15011,15014,15015,15021,15024,15026-15040,15043-
15045,15047,15048,16000-16009,16020-16024,17000,17007,17024,18000-
18002,18011,18012,18024,18025,20000,20001,21000-21007,26004-26006,26008-26010,30000-
30010,30012,31001-31004,31007-31012
'6: 1009,2010-2014,2021,5002,5008,5009,5013,5017,5018,6003,6004,6016,6037,7002,7006-
7008,10031,10034,10037,10066,10067,13013,15004,15006,15015,15024,15027,15036,15039,180
03,18010,18013,18014,18017-18020,18031,23000-23031
'7: 2007,5016,5021,6016,8001,11000,11006,11010,11011,13000-13002,15045,17020-17023
'8: 2002,2017,3003,3010,3011,4000,5015,5021,6003,6009,7001-
7003,7006,8001,10008,10025,12000-12009,12011-12036,12040-12052,12060-12062,12070-
12072,13005,13012,13013,14001-14003,14005,15001,15008,15009,15021,15043,15045,16006-
16008,16020-16023,17004,18000,18001,18007-18015,18017-18020,18022-18031,20001,20010-
20016,20020,20022-20024,20030,22000-22022,23000-23031,24000-24020,24022,24025-
24027,25000-25023,26004,27002,29008-29019,29021-29023,30001,30002,30005,31004,31006
'9: 2005,2018,2019,3007,3012,5009,5011,5013,5014,5018,5020,5023,6015,6027,8000-
8012,8016,8017,8019,8020,8023-8036,8038-8044,8046,8049-
8051,8053,8054,10001,10014,10022-
10024,10026,10035,10042,10057,13011,14007,15015,15027,15029,15034-
15036,15039,15040,15047,15048,16024,17024,18016,20001
'10: 3000-3003,3006-3012,13001
'11: 27000-27025,28000-28002,28005,28006,28010-28043,28080-28092,28150-28169,28173-
28176,28200,28201,28250-28256,29000-29003,29008-29019,29021-29023
'12: 24000-24020,24022,24025,24026,24027,25000-25023,31001-31003,31005,31007-
31010,31012
'13: 5006,5007,5020,15038,21001-21005

'Set up defaults
Options.Prediction_LineColor = vbRed
Options.Prediction_ColumnColor = &HFF8080 'light blue
Options.SaveLoad = vbChecked

Call ReadOptionsFromDB
Call ReadOptionsFromFile

```

```

gsDefaultConnString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=%PATH%;Persist
Security Info=False"
fmMain.Show

End Sub

Public Sub ReadOptionsFromDB()

    On Error GoTo ErrorHandler

    Dim TempConn As New ADODB.Connection
    Dim TempRS As New ADODB.Recordset, TempRS1 As New ADODB.Recordset
    Dim lLevel As Long, lLastCatNum As Long, lTemp As Long
    Dim lNum As Long, lNumCount As Long
    Dim a As New ADOX.Catalog

    gbDBDetected = False

    lLevel = 0
    gsConnectionStringToMainDB = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" &
App.Path & "\VF_DB.mdb;Persist Security Info=False"
    'gsConnectionStringToMainDB = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" &
App.Path & "\VF_DBL.mdb;Persist Security Info=False"

    'Open connection to DB
    lLevel = 1
    TempConn.ConnectionString = gsConnectionStringToMainDB
    TempConn.Open

    gbDBDetected = True

    '*****READ CUSTOM CATEGORIES*****
    'Open recordset
    lLevel = 2
    Call OpenDBTable(TempConn, TempRS, "SELECT * FROM HVC")
    lNum = 0
    ReDim CustomCategoryInfo(lNum)
    Do While Not TempRS.EOF
        lNum = lNum + 1
        If lNum = 1 Then
            ReDim CustomCategoryInfo(1 To lNum)
        Else
            ReDim Preserve CustomCategoryInfo(1 To lNum)
        End If

        CustomCategoryInfo(lNum).Number = TempRS!HVC_Number
        CustomCategoryInfo(lNum).Description = "" & TempRS!HVC_Name

        TempRS.MoveNext
    Loop
    glCustomCategoriesNum = lNum

    'Close recordset
    TempRS.Close
    '*****READ CUSTOM CATEGORIES*****

    '*****READ SW Packages*****
    'Open recordset
    lLevel = 5
    Call OpenDBTable(TempConn, TempRS, "SELECT * FROM SWPackage")
    lNum = 0
    ReDim SWPackageInfo(lNum)
    Do While Not TempRS.EOF
        lNum = lNum + 1
        If lNum = 1 Then
            ReDim SWPackageInfo(1 To lNum)
        Else
            ReDim Preserve SWPackageInfo(1 To lNum)
        End If

        SWPackageInfo(lNum).Number = TempRS!Package_ID
        SWPackageInfo(lNum).Name = "" & TempRS!Package_Name
        SWPackageInfo(lNum).VulnDB.DBPathName = "" & TempRS!Vuln_DB_PathName
        SWPackageInfo(lNum).VulnDB.TableName = "" & TempRS!Vuln_DB_TableName
        SWPackageInfo(lNum).VulnDB.FieldName_ID = "" & TempRS!Vuln_DB_FieldName_ID
        SWPackageInfo(lNum).VulnDB.FieldName_Description = "" &
TempRS!Vuln_DB_FieldName_Description

```

```

SWPackageInfo(lNum).ScanningDB.TableName = "" & TempRS!Scan_DB_TableName
SWPackageInfo(lNum).ScanningDB.FieldName = "" & TempRS!Scan_DB_FieldName

TempRS.MoveNext
Loop
'glCustomCategoriesNum = lNum

'Close recordset
TempRS.Close
'*****READ SW Packages*****

'*****READ CYBERCOP CATEGORIES*****
'Open recordset
lLevel = 3
lLastCatNum = -1
Call OpenDBTable(TempConn, TempRS, "SELECT * FROM VulnerabilityCategory",
"InternalPackageID ASC, VulnerabilityCategory_Number ASC")
lNum = 0
'ReDim SWPackageInfo(lNum)
If Not TempRS.EOF Then
Do While Not TempRS.EOF
If TempRS!InternalPackageID <> lLastCatNum Then
lNum = lNum + 1
'If lNum = 1 Then
'ReDim SWPackageInfo(1 To lNum)
'Else
'ReDim Preserve SWPackageInfo(1 To lNum)
'End If

lLastCatNum = TempRS!InternalPackageID

lNumCount = 0
'ReDim SWPackageInfo(lNum).VulnDB.MainCategories(lNumCount)
End If

lNumCount = lNumCount + 1
If lNumCount = 1 Then
'ReDim SWPackageInfo(lNum).VulnDB.MainCategories(1 To lNumCount)
Else
'ReDim Preserve SWPackageInfo(lNum).VulnDB.MainCategories(1 To lNumCount)
End If

SWPackageInfo(lNum).VulnDB.MainCategories(lNumCount).Number =
TempRS!VulnerabilityCategory_ID
SWPackageInfo(lNum).VulnDB.MainCategories(lNumCount).ID =
TempRS!VulnerabilityCategory_Number
SWPackageInfo(lNum).VulnDB.MainCategories(lNumCount).Description =
TempRS!VulnerabilityCategory_Description
SWPackageInfo(lNum).VulnDB.MainCategories(lNumCount).Sequence = ""

SQL = "SELECT * FROM Vulnerability WHERE VulnerabilityCategory_Number=" &
CStr(TempRS!VulnerabilityCategory_Number)
Call OpenDBTable(TempConn, TempRS1, SQL, "Vulnerability_Number")
Do While Not TempRS1.EOF
If SWPackageInfo(lNum).VulnDB.MainCategories(lNumCount).Sequence = "" Then
SWPackageInfo(lNum).VulnDB.MainCategories(lNumCount).Sequence =
TempRS1!Vulnerability_Number
Else
SWPackageInfo(lNum).VulnDB.MainCategories(lNumCount).Sequence =
SWPackageInfo(lNum).VulnDB.MainCategories(lNumCount).Sequence & "," &
TempRS1!Vulnerability_Number
End If
TempRS1.MoveNext
Loop
TempRS1.Close

TempRS.MoveNext
Loop

'*****READ MAPPING TABLES*****
'Open recordset
lLevel = 4
lLastCatNum = -1
Call OpenDBTable(TempConn, TempRS1, "SELECT * FROM Mapping", "HVC_Number ASC,
Vulnerability_Number ASC")
lNumCount = 0
'ReDim SWPackageInfo(lNum).VulnDB.Mapping(0)

```

```

Do While Not TempRS1.EOF
    lNumCount = 0
    lTemp = CLng(TempRS1!InternalPackageID)
    If lTemp <= UBound(SWPackageInfo) Then

        If lLastCatNum <> TempRS1!HVC_Number Then
            lLastCatNum = TempRS1!HVC_Number
            If lLastCatNum = 1 Then
                ReDim SWPackageInfo(lTemp).VulnDB.Mapping(1 To lLastCatNum)
            Else
                ReDim Preserve SWPackageInfo(lTemp).VulnDB.Mapping(1 To lLastCatNum)
            End If
            SWPackageInfo(lTemp).VulnDB.Mapping(lLastCatNum) = ""
        End If

        If SWPackageInfo(lTemp).VulnDB.Mapping(lLastCatNum) = "" Then
            SWPackageInfo(lTemp).VulnDB.Mapping(lLastCatNum) =
TempRS1!Vulnerability_Number
        Else
            SWPackageInfo(lTemp).VulnDB.Mapping(lLastCatNum) =
SWPackageInfo(lTemp).VulnDB.Mapping(lLastCatNum) & "," & TempRS1!Vulnerability_Number
        End If
    End If

    TempRS1.MoveNext
Loop
'Close recordset
TempRS1.Close
*****READ MAPPING TABLES*****
End If

'Close recordset
TempRS.Close
*****READ CYBERCOP CATEGORIES*****

'Close connection to DB
TempConn.Close

Quit:
Exit Sub

ErrorHandler:
Select Case lLevel
    Case 1
        'Can't open connection to DB
        Call MsgBox("Can't open database")
        End
    Case Else
        Resume Next
End Select
Resume Next
Resume

End Sub

Public Function OpenDBTable(OpenConn As ADODB.Connection, OpenRecordset As
ADODB.Recordset, sOpenTable As String, Optional sOrderBy As String = "", Optional
CurLoc As CursorLocationEnum = adUseServer, Optional CurType As CursorTypeEnum =
adOpenKeyset, Optional LckType As LockTypeEnum = adLockOptimistic) As Boolean

    On Error GoTo Quit

    OpenDBTable = False

    Set OpenRecordset = New ADODB.Recordset
    OpenRecordset.CursorLocation = CurLoc
    OpenRecordset.CursorType = CurType
    OpenRecordset.LockType = LckType
    If sOrderBy = "" Then
        OpenRecordset.Open sOpenTable, OpenConn
    Else
        OpenRecordset.Open sOpenTable & " ORDER BY " & sOrderBy, OpenConn
    End If

    OpenDBTable = True

Quit:

```

```

Exit Function
End Function
Public Sub ReadOptionsFromFile()

Dim FileNum As Long, Place As Long
Dim InputData As String, sLeftHS As String, sRightHS As String
Dim lNum As Long, lNumCount As Long
Dim bRedim As Boolean

'Open file
FileNum = FreeFile

'Create text file if does not exist
If Dir(App.Path & "\VulnPredict.vpl", vbArchive + vbHidden + vbNormal + vbReadOnly +
vbSystem + vbVolume) <> "VulnPredict.vpl" Then
    Call CreateDefaultTextFile
End If

'Open file for reading
Open (App.Path & "\VulnPredict.vpl") For Input As #FileNum

Do While Not EOF(FileNum) ' Check for end of file.
    Line Input #FileNum, InputData ' Read line of data.

    Select Case InputData

        Case "DATABASE PATHS"
            lNum = 0
            ReDim gsPreviousPaths(0)
            Do

                'Read Next line
                Line Input #FileNum, InputData

                If InputData <> "END OF DATABASE PATHS" Then
                    lNum = lNum + 1
                    If lNum = 1 Then
                        ReDim gsPreviousPaths(1 To lNum)
                    Else
                        ReDim Preserve gsPreviousPaths(1 To lNum)
                    End If
                    gsPreviousPaths(lNum) = InputData
                End If

            Loop While InputData <> "END OF DATABASE PATHS"

        Case "ADJECTIVE LIST"
            lNum = 0
            ReDim gsAdjectives(0)
            Do

                'Read Next line
                Line Input #FileNum, InputData

                If InputData <> "END OF ADJECTIVE LIST" Then
                    lNum = lNum + 1
                    If lNum = 1 Then
                        ReDim gsAdjectives(1 To lNum)
                    Else
                        ReDim Preserve gsAdjectives(1 To lNum)
                    End If
                    gsAdjectives(lNum) = InputData
                End If

            Loop While InputData <> "END OF ADJECTIVE LIST"

        Case "COMPARISON DATABASE PATHS"
            lNum = 0
            ReDim gsPreviousComparePaths(0)
            Do

                'Read Next line
                Line Input #FileNum, InputData

```

```

    If InputData <> "END OF COMPARISON DATABASE PATHS" Then
        lNum = lNum + 1
        If lNum = 1 Then
            ReDim gsPreviousComparePaths(1 To lNum)
        Else
            ReDim Preserve gsPreviousComparePaths(1 To lNum)
        End If
        gsPreviousComparePaths(lNum) = InputData
    End If

    Loop While InputData <> "END OF COMPARISON DATABASE PATHS"

    Case "OPTIONS"
    Do

        'Read Next line
        Line Input #FileNum, InputData

        If InputData <> "END OF OPTIONS" Then
            Place = InStr(1, InputData, "|||")
            If Place > 0 Then
                sLeftHS = Left$(InputData, Place - 1)
                sRightHS = Mid$(InputData, Place + 3)
                If IsNumeric(sRightHS) Then
                    Select Case sLeftHS
                        Case "Prediction_LineColor": Options.Prediction_LineColor =
                            CLng(sRightHS)
                        Case "Prediction_ColumnColor": Options.Prediction_ColumnColor =
                            CLng(sRightHS)
                        Case "SaveLoad": Options.SaveLoad = CInt(sRightHS)
                    End Select
                End If
            End If
        End If

        Loop While InputData <> "END OF OPTIONS"

    End Select
    Loop

    'Close file
    Close #FileNum 'Close file.

End Sub

Private Sub CreateDefaultTextFile()

    Dim FileNum As Long, Count As Long
    Dim InputData As String

    FileNum = FreeFile

    'Create file
    Open (App.Path & "\VulnPredict.vpl") For Output Access Write As #FileNum

    'Create default entries
    Print #FileNum, "DATABASE PATHS"
    Print #FileNum, "END OF DATABASE PATHS"

    Print #FileNum, "ADJECTIVE LIST"
    Print #FileNum, "Almost"
    Print #FileNum, "More or less"
    Print #FileNum, "More than"
    Print #FileNum, "Much more than"
    Print #FileNum, "END OF ADJECTIVE LIST"

    Print #FileNum, "COMPARISON DATABASE PATHS"
    Print #FileNum, "END OF COMPARISON DATABASE PATHS"

    Print #FileNum, "OPTIONS"
    Print #FileNum, "Prediction_LineColor|||" & Options.Prediction_LineColor
    Print #FileNum, "Prediction_ColumnColor|||" & Options.Prediction_ColumnColor
    Print #FileNum, "SaveLoad|||" & Options.SaveLoad
    Print #FileNum, "END OF OPTIONS"

End Sub

```



```

Public Sub WriteOptionsToFile()

    On Error GoTo ErrorHandler

    Dim FileNum As Long, Count As Long, CountIn As Long, lNum As Long
    Dim lLine As Long, lTempNum As Long

    'Open file
    lLine = 0
    FileNum = FreeFile
    lLine = 1

    'Destroy previous version on file
    If Dir(App.Path & "\VulnPredict.vpl", vbArchive + vbHidden + vbNormal + vbReadOnly +
vbSystem + vbVolume) = "VulnPredict.vpl" Then
        lLine = 2
        Kill App.Path & "\VulnPredict.vpl"
    End If

    'Create file
    lLine = 3
    Open (App.Path & "\VulnPredict.vpl") For Output Access Write As #FileNum

    'Write entries
    '*****
    Print #FileNum, "DATABASE PATHS"
    '*****
    lLine = 4
    lNum = UBound(gsPreviousPaths)
    lLine = 5
    For Count = 1 To lNum
        Print #FileNum, gsPreviousPaths(Count)
    Next Count

    Print #FileNum, "END OF DATABASE PATHS"

    '*****
    Print #FileNum, "ADJECTIVE LIST"
    '*****
    lLine = 6
    lNum = UBound(gsAdjectives)
    lLine = 7
    For Count = 1 To lNum
        Print #FileNum, gsAdjectives(Count)
    Next Count

    Print #FileNum, "END OF ADJECTIVE LIST"

    '*****
    Print #FileNum, "COMPARISON DATABASE PATHS"
    '*****
    lLine = 8
    lNum = UBound(gsPreviousComparePaths)
    lLine = 9
    For Count = 1 To lNum
        Print #FileNum, gsPreviousComparePaths(Count)
    Next Count

    Print #FileNum, "END OF COMPARISON DATABASE PATHS"

    '*****
    Print #FileNum, "OPTIONS"
    '*****
    Print #FileNum, "Prediction_LineColor|||" & Options.Prediction_LineColor
    Print #FileNum, "Prediction_ColumnColor|||" & Options.Prediction_ColumnColor
    Print #FileNum, "SaveLoad|||" & Options.SaveLoad
    Print #FileNum, "END OF OPTIONS"

    lLine = 12
    Close #FileNum

Quit:
Exit Sub

```

```

ErrorHandler:
Select Case lLine
Case 4, 6, 8, 10: lNum = 0: Resume Next
Case 20: lTempNum = 0: Resume Next
Case Else: Resume Quit
End Select
Resume

End Sub

Private Sub SplitStringIntoLeftRight(sStringToSplit As String, sLeft As String, sRight
As String, Optional sSeparator As String = "|||")

Dim Place As Long

Place = InStr(1, sStringToSplit, sSeparator)
If Place > 0 Then
sLeft = Left$(sStringToSplit, Place - 1)
sRight = Mid$(sStringToSplit, Place + Len(sSeparator))
Else
sLeft = sStringToSplit
sRight = ""
End If

End Sub

Public Sub OptimizeCommaSeperatedNumbers(sString As String)

Dim lFirstNumberInSequence As Long, lNextNumberInSequence As Long
Dim PlaceS As Long, PlaceNextS As Long, PlaceNextE As Long, PlaceTemp As Long
Dim lSequenceStartComma As Long, lSequenceEndComma As Long
Dim sTemp As String, s1 As String, sF As String
Dim lNumInSequence As Long, Count As Long
Dim bOK As Boolean
Dim sSequences() As String
Dim lNumSequences As Long

'Remove sequences
PlaceS = InStr(1, sString, "-")
Do While (PlaceS > 0)
PlaceNextS = InStrRev(sString, ",", PlaceS)
If PlaceNextS > 0 Then
sTemp = Mid$(sString, PlaceNextS + 1, PlaceS - (PlaceNextS + 1))
Else
sTemp = Left$(sString, PlaceS - 1)
End If

If IsNumeric(sTemp) Then
sF = sTemp
s1 = sTemp & "-"
lFirstNumberInSequence = CLng(sTemp)

PlaceNextS = InStr(PlaceS, sString, ",")
If PlaceNextS > 0 Then
sTemp = Mid$(sString, PlaceS + 1, PlaceNextS - (PlaceS + 1))
Else
sTemp = Mid$(sString, PlaceS + 1)
End If

If IsNumeric(sTemp) Then
s1 = s1 & sTemp
lNextNumberInSequence = CLng(sTemp)

Do While lFirstNumberInSequence < lNextNumberInSequence
lFirstNumberInSequence = lFirstNumberInSequence + 1
sF = sF & "," & CStr(lFirstNumberInSequence)
Loop

sString = Replace(sString, s1, sF)
End If
End If

PlaceS = InStr(PlaceS + 1, sString, "-")
Loop

bOK = True

```

```

lNumSequences = 0
PlaceS = 0
Do While bOK
    lNumInSequence = 1
    lFirstNumberInSequence = GetNextNumber(PlaceS + 1, sString, PlaceNextS)
    lNextNumberInSequence = GetNextNumber(PlaceNextS + 1, sString, PlaceNextE)

    If ((lFirstNumberInSequence + 1) = lNextNumberInSequence) Then
        lSequenceStartComma = PlaceNextS
        Do While ((lFirstNumberInSequence + 1) = lNextNumberInSequence)
            lNumInSequence = lNumInSequence + 1

            lFirstNumberInSequence = lNextNumberInSequence
            lNextNumberInSequence = GetNextNumber(PlaceNextE + 1, sString, PlaceTemp)

            If ((lFirstNumberInSequence + 1) = lNextNumberInSequence) Then
                lSequenceEndComma = PlaceNextE

                PlaceNextE = PlaceTemp
                PlaceS = InStr(PlaceS + 1, sString, ",")
            Loop
        Else
            PlaceS = InStr(PlaceS + 1, sString, ",")
            If PlaceNextE < 1 Then PlaceS = 0
        End If

        If (lNumInSequence > 2) Then
            lNumSequences = lNumSequences + 1
            If lNumSequences = 1 Then
                ReDim sSequences(1 To lNumSequences)
            Else
                ReDim Preserve sSequences(1 To lNumSequences)
            End If
            sSequences(lNumSequences) = Mid$(sString, lSequenceStartComma, lSequenceEndComma - lSequenceStartComma + 1)
        End If

        bOK = (PlaceS > 0)

    Loop

    If lNumSequences > 0 Then
        For Count = 1 To lNumSequences
            sString = Replace(sString, sSequences(Count), "-")
        Next Count
    End If

End Sub

Private Function GetNextNumber(lFromComma As Long, sString As String, lNextComma As Long) As Long

    Dim lTemp As Long
    Dim sTemp As String

    lNextComma = InStr(lFromComma, sString, ",")
    If lNextComma > 0 Then
        sTemp = Mid$(sString, lFromComma, lNextComma - lFromComma)
        If IsNumeric(sTemp) Then
            lTemp = CLng(sTemp)
        End If
    Else
        sTemp = Mid$(sString, lFromComma)
        If sTemp <> "" Then
            If IsNumeric(sTemp) Then lTemp = CLng(sTemp)
        End If
    End If
    GetNextNumber = lTemp

End Function

Public Sub CreateCommaSeperatedNumbersFromOptimizedString(sString As String)

    Dim PlaceS As Long, PlaceNextS As Long, lFirstNumberInSequence As Long,
    lNextNumberInSequence As Long
    Dim sTemp As String, sF As String, sI As String

```

```

PlaceS = InStr(1, sString, "-")
Do While (PlaceS > 0)
  PlaceNextS = InStrRev(sString, ",", PlaceS)
  If PlaceNextS > 0 Then
    sTemp = Mid$(sString, PlaceNextS + 1, PlaceS - (PlaceNextS + 1))
  Else
    sTemp = Left$(sString, PlaceS - 1)
  End If

  If IsNumeric(sTemp) Then
    sF = sTemp
    s1 = sTemp & "-"
    lFirstNumberInSequence = CLng(sTemp)

    PlaceNextS = InStr(PlaceS, sString, ",")
    If PlaceNextS > 0 Then
      sTemp = Mid$(sString, PlaceS + 1, PlaceNextS - (PlaceS + 1))
    Else
      sTemp = Mid$(sString, PlaceS + 1)
    End If

    If IsNumeric(sTemp) Then
      s1 = s1 & sTemp
      lNextNumberInSequence = CLng(sTemp)

      Do While lFirstNumberInSequence < lNextNumberInSequence
        lFirstNumberInSequence = lFirstNumberInSequence + 1
        sF = sF & "," & CStr(lFirstNumberInSequence)
      Loop

      sString = Replace(sString, s1, sF)
    End If
  End If

  PlaceS = InStr(PlaceS + 1, sString, "-")
Loop

End Sub

```

```

Private Function CheckFile(Path As String) As Boolean

  On Error GoTo ErrorHandler

  Dim Result As Boolean

  Result = False

  If Dir(Path) <> "" Then
    Result = True
  End If

Quit:

  CheckFile = Result

  Exit Function

ErrorHandler:

  Resume Quit

End Function

```

```

Public Function GetProcAddress(Address As Long) As Long
  GetProcAddress = Address
End Function

```

```

Public Function BrowseCallbackProc(ByVal lhWnd As Long, ByVal luMsg As Long, ByVal
lParam As Long, ByVal lpData As Long) As Long
  On Error Resume Next

  Dim Path As String
  Dim sBuffer As String

  Select Case luMsg

    Case BFFM_INITIALIZED '1

```



```

If InitDir <> "" Then
    Call SendMessageStr(lhWnd, BFFM_SETSELECTIONA, 1, lstrcat(InitDir, ""))
End If

Case BFFM_SELCHANGED '2
    If (lParam) Then
        sBuffer = Space(MAX_PATH)
        SHGetPathFromIDList lParam, sBuffer
        Path = Left(sBuffer, InStr(sBuffer, vbNullChar))
    End If

    If Path <> Chr(0) Then

        If gsFilter <> "" Then
            If CheckFile(Left(Path, InStr(Path, vbNullChar) - 1) & "\" & gsFilter) Then
                Call SendMessageStr(lhWnd, BFFM_SETSTATUSTEXTA, 0, gsStatusTextFound)
                Call PostMessage(lhWnd, BFFM_ENABLEOK, 0, 1)
            Else
                Call SendMessageStr(lhWnd, BFFM_SETSTATUSTEXTA, 0, Path)
                Call PostMessage(lhWnd, BFFM_ENABLEOK, 0, 0)
            End If
        Else
            Call SendMessageStr(lhWnd, BFFM_SETSTATUSTEXTA, 0, Path)
        End If

        Else

        If gsFilter <> "" Then
            Call SendMessageStr(lhWnd, BFFM_SETSTATUSTEXTA, 0, Path)
            Call PostMessage(lhWnd, BFFM_ENABLEOK, 0, 0)
        Else
            Call SendMessageStr(lhWnd, BFFM_SETSTATUSTEXTA, 0, Path)
        End If

        End If

    Case BFFM_VALIDATEFAILEDA '3

    Case BFFM_VALIDATEFAILEDW = 4 '4

End Select
End Function

```

```

Public Function GetNumberOfDBsInDirectory(sDirectory As String, DirFiles() As
ScanInfoType) As Long

    Dim sTempDir As String, sTempFile As String
    Dim lNum As Long

    sTempDir = sDirectory
    If sTempDir = "" Then
        GetNumberOfDBsInDirectory = 0
        Exit Function
    End If

    lNum = 0
    'Get first file name in directory
    sTempFile = Dir(sTempDir, vbArchive + vbHidden + vbNormal + vbReadOnly + vbSystem +
vbVolume)

    'Loop through all files in directory
    Do While sTempFile <> ""
        If (sTempFile <> ".") And (sTempFile <> "..") Then
            'If file has .mdb extension -> increment counter
            If LCase(Right$(sTempFile, 4)) = ".mdb" Then
                lNum = lNum + 1
                If lNum = 1 Then
                    ReDim DirFiles(1 To lNum)
                Else
                    ReDim Preserve DirFiles(1 To lNum)
                End If

                DirFiles(lNum).TableName = sTempFile
            End If
        End If

        'Get next file
    
```

```

    sTempFile = Dir
Loop

GetNumberOfDBsInDirectory = lNum

End Function

```

```

Public Sub GetNumberOfVulnerabilitiesForTable(sTablePath As String, TableInfo As ScanInfoType)

    On Error GoTo ErrorHandler

    Dim Count As Long, CountIn As Long
    Dim TempConn As New ADODB.Connection
    Dim TempRS As New ADODB.Recordset
    Dim sGUIDTableName As String
    Dim lIndex() As Long

    ReDim TableInfo.VulnCount(1 To glNumVulnerabilityCategories)
    ReDim TableInfo.VulnID(1 To glNumVulnerabilityCategories)

    'Open ADO connection to database
    TempConn.ConnectionString = Replace(gsDefaultConnString, "%PATH%", sTablePath & TableInfo.TableName)
    TempConn.Open

    sGUIDTableName = SWPackageCurrent.ScanningDB.TableName

    'Open recordset
    TempRS.CursorLocation = adUseServer
    TempRS.CursorType = adOpenStatic
    TempRS.LockType = adLockOptimistic

    'Get data
    TempRS.Open "SELECT * FROM " & sGUIDTableName & " WHERE " & SWPackageCurrent.ScanningDB.FieldName & ">0 ORDER BY " & SWPackageCurrent.ScanningDB.FieldName & " ASC", TempConn, adCmdTable

    If fmMain.chkMap.Value = vbChecked Then
        For Count = 1 To glNumVulnerabilityCategories
            TableInfo.VulnCount(Count) = Count
        Next Count
    Else
        For Count = 1 To glNumVulnerabilityCategories
            TableInfo.VulnID(Count) = SWPackageCurrent.VulnDB.MainCategories(Count).Number
        Next Count
    End If

    Do While Not TempRS.EOF
        If fmMain.chkMap.Value = vbChecked Then
            Call MapVulnerabilityToCategory(TempRS.Fields(SWPackageCurrent.ScanningDB.FieldName).Value, TableInfo)
        Else
            'This function should only return one, but can return more
            lIndex = SubCategoryIDBelongsToWhichCategory(TempRS.Fields(SWPackageCurrent.ScanningDB.FieldName).Value)

            For Count = 1 To UBound(lIndex)
                For CountIn = 1 To glNumVulnerabilityCategories
                    If TableInfo.VulnID(CountIn) = lIndex(Count) Then
                        TableInfo.VulnCount(CountIn) = TableInfo.VulnCount(CountIn) + 1
                        GoTo AfterCountIn
                    End If
                Next CountIn
            AfterCountIn:
            Next Count
        End If

        TempRS.MoveNext
    Loop

    TempRS.Close

Quit:

```

```

Exit Sub

ErrorHandler:
Resume Quit

End Sub

Public Sub MapVulnerabilityToCategory(VulnID As Long, TableInfo As ScanInfoType)

On Error GoTo ErrorHandler

Dim Count As Long, CountArray As Long, lVuln As Long, Place As Long
Dim sFirst As String, sLast As String
Dim sCatArray() As String
Dim sTemp As String
Dim bFound As Boolean

bFound = False
'If lLastBlah <> VulnID Then
' lLastBlah = VulnID
' Stop
'End If
For Count = 1 To glNumVulnerabilityCategories
sTemp = SWPackageCurrent.VulnDB.Mapping(Count)
Call CreateCommaSeperatedNumbersFromOptimizedString(sTemp)
sCatArray = Split(sTemp, ",")

For CountArray = 0 To UBound(sCatArray)
If IsNumeric(sCatArray(CountArray)) Then
lVuln = CLng(sCatArray(CountArray))
If lVuln > VulnID Then GoTo BeforeNext
If VulnID = lVuln Then
'Found category
TableInfo.VulnCount(Count) = TableInfo.VulnCount(Count) + 1
bFound = True
End If
End If
Next CountArray

BeforeNext:
Next Count

Quit:
'If Not bFound Then Stop
Exit Sub

ErrorHandler:
Resume Next
Resume

End Sub

Public Function SubCategoryIDBelongsToWhichCategory(SubCatID As Long) As Long()

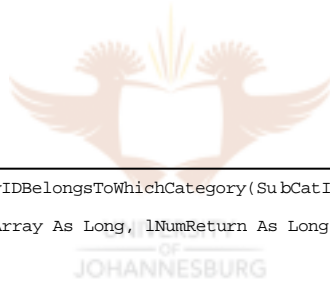
Dim Count As Long, CountArray As Long, lNumReturn As Long
Dim sTemp As String
Dim sSplit() As String
Dim lReturn() As Long

ReDim lReturn(0)
lNumReturn = 0
For Count = 1 To UBound(SWPackageCurrent.VulnDB.MainCategories)
sTemp = SWPackageCurrent.VulnDB.MainCategories(Count).Sequence
Call CreateCommaSeperatedNumbersFromOptimizedString(sTemp)

If sTemp <> "" Then
sSplit = Split(sTemp, ",")

For CountArray = 0 To UBound(sSplit)
If CLng(sSplit(CountArray)) > SubCatID Then GoTo AfterCountArray
If CLng(sSplit(CountArray)) = SubCatID Then
lNumReturn = lNumReturn + 1
If lNumReturn = 1 Then
ReDim lReturn(1 To lNumReturn)
Else
ReDim Preserve lReturn(1 To lNumReturn)
End If

```



```

        lReturn(lNumReturn) = SWPackageCurrent.VulnDB.MainCategories(Count).Number
        GoTo AfterCountArray
    End If
    Next CountArray
End If
AfterCountArray:
Next Count

```

```
SubCategoryIDBelongsToWhichCategory = lReturn
```

```
End Function
```

```
Public Sub UnloadApplication(Optional UnloadFmMain As Boolean = True)
```

```
    Call WriteOptionsToFile
```

```

    If gbFmCalculationsLoaded Then Unload fmCalculations
    If gbFmGraphicsLoaded Then Unload fmGraphics
    If gbFmHelpLoaded Then Unload fmHelp
    If gbFmOptionsLoaded Then Unload fmOptions
    If gbFmSetupLoaded Then Unload fmSetup
    If gbFmSetupNames Then Unload fmSetupNames
    If gbFmSWSetupLoaded Then Unload fmSWSetup
    If UnloadFmMain Then Unload fmMain

```

```
End Sub
```

```
Public Sub SetTab(ByRef tsTabStrip As TabStrip, ByRef frmFrameArray As Object)
```

```
    Dim tabIndex As Long
```

```

    'Go through all of the frames, only setting the wanted one visible
    For tabIndex = 1 To frmFrameArray.Count
        If tabIndex = tsTabStrip.SelectedItem.Index Then
            'If this is the selected tab, set the frame visible
            frmFrameArray(tabIndex - 1).Visible = True
            'frmFrameArray(tabIndex - 1).ZOrder = 0
        Else
            'Hide unwanted frames
            frmFrameArray(tabIndex - 1).Visible = False
        End If
    Next tabIndex

```

```
End Sub
```

```
Public Sub ClearSWPackageInfo(TempSWPackageInfo As SWPackageInfoType)
```

```

    TempSWPackageInfo.Number = 0
    TempSWPackageInfo.Name = "<New Package>"

    TempSWPackageInfo.VulnDB.DBPathName = ""
    TempSWPackageInfo.VulnDB.TableName = ""
    TempSWPackageInfo.VulnDB.FieldName_ID = ""
    TempSWPackageInfo.VulnDB.FieldName_Description = ""
    ReDim TempSWPackageInfo.VulnDB.Mapping(0)

    TempSWPackageInfo.ScanningDB.TableName = ""
    TempSWPackageInfo.ScanningDB.FieldName = ""
    TempSWPackageInfo.ScanningDB.SampleDB.DBPathName = ""
    TempSWPackageInfo.ScanningDB.SampleDB.CreatedFrom = 0
    TempSWPackageInfo.ScanningDB.SampleDB.SQLResultTableName = ""

```

```
End Sub
```

```
Public Function GetMaxID(sTable As String, sIDField As String) As Long
```

```
    On Error GoTo ErrorHandler
```

```

    Dim TempConn As New ADODB.Connection
    Dim TempRS As New ADODB.Recordset
    Dim lMax As Long

```

```

    'Open connection to DB
    TempConn.ConnectionString = gsConnectionStringToMainDB
    TempConn.Open

```

```
    'Open recordset
```



```

TempRS.CursorLocation = adUseServer
TempRS.CursorType = adOpenKeyset
TempRS.LockType = adLockOptimistic
TempRS.Open "SELECT MAX(" & sIDField & ") AS MaxID FROM " & sTable, TempConn
lMax = TempRS!MaxID

Quit:
  GetMaxID = lMax
  Exit Function

ErrorHandler:
  lMax = 0
  Resume Quit

End Function

```

B.3 SOURCE CODE FOR THE CONTROLS

B.3.1 The “GraphView” control

The “GraphView” control is used to create any graph that needs to be displayed.

The source code for this control follows below.

```

Option Explicit

Dim bPropertiesRead As Boolean

Private bColumnBarsAlreadyLoaded As Boolean
Private bDoPicture As Boolean

Private Const m_const_Font_Size As Long = 10
Private Const mX_const_Min As Long = 0
Private mX_const_Max As Long '= glnumVulnerabilityCategories
Private Const mX_const_Increment As Long = 1
Private Const mY_const_Min As Long = 0
Private Const mY_const_Max As Long = 100
Private Const mY_const_Increment As Long = 10

Private mHeading As String
Private mFont_Size As Long
Private mPrediction_DisplayInfo As Boolean
Private mPrediction_Heading_Bottom As String
Private mPrediction_Heading_Top As String
Private mPrediction_LineColor As OLE_COLOR
Private mSpecial_LineColor As OLE_COLOR
Private mSpecial_Display As Boolean
Private mSpecial_Lowerbound As Long
Private mSpecial_Upperbound As Long
Private mUseCustomMouseIcon As Boolean
Private mX_Heading As String
Private mX_Min As Long
Private mX_Max As Long
Private mX_Increment As Long
Private mX_Values As String
Private mY_Heading As String
Private mY_Min As Long
Private mY_Max As Long
Private mY_Increment As Long
Private mY_GridLines As Boolean

Private Type PredType
  Lowerbound As Long
  Upperbound As Long
End Type

```

```

Public Event MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
Public Event Click()

Private Sub lblColumnVal_Click(Index As Integer)
    RaiseEvent Click
End Sub

Private Sub lblColumnVal_MouseMove(Index As Integer, Button As Integer, Shift As
Integer, X As Single, Y As Single)
    RaiseEvent MouseMove(Button, Shift, X, Y)
    Call DoCustomCursorIcon
End Sub

Private Sub lblHeading_Click()
    RaiseEvent Click
End Sub

Private Sub lblHeading_MouseMove(Button As Integer, Shift As Integer, X As Single, Y
As Single)
    RaiseEvent MouseMove(Button, Shift, X, Y)
    Call DoCustomCursorIcon
End Sub

Private Sub lblHeadingX_Click()
    RaiseEvent Click
End Sub

Private Sub lblHeadingX_MouseMove(Button As Integer, Shift As Integer, X As Single, Y
As Single)
    RaiseEvent MouseMove(Button, Shift, X, Y)
    Call DoCustomCursorIcon
End Sub

Private Sub lblXValue_Click(Index As Integer)
    RaiseEvent Click
End Sub

Private Sub lblXValue_MouseMove(Index As Integer, Button As Integer, Shift As Integer,
X As Single, Y As Single)
    RaiseEvent MouseMove(Button, Shift, X, Y)
    Call DoCustomCursorIcon
End Sub

Private Sub lblYValue_Click(Index As Integer)
    RaiseEvent Click
End Sub

Private Sub lblYValue_MouseMove(Index As Integer, Button As Integer, Shift As Integer,
X As Single, Y As Single)
    RaiseEvent MouseMove(Button, Shift, X, Y)
    Call DoCustomCursorIcon
End Sub

Private Sub UserControl_Click()
    RaiseEvent Click
End Sub

Private Sub UserControl_InitProperties()
    mX_const_Max = g1NumVulnerabilityCategories
    mHeading = "Heading"
    mFont_Size = m_const_Font_Size
    mSpecial_LineColor = vbWindowText
    mPrediction_DisplayInfo = False
    mPrediction_Heading_Bottom = ""
    mPrediction_Heading_Top = ""
    mPrediction_LineColor = vbWindowText
    mSpecial_Display = False
    mSpecial_Lowerbound = 0
    mSpecial_Upperbound = 0
    mUseCustomMouseIcon = False
    mX_Heading = "X-Axis"
    mX_Min = mX_const_Min
    mX_Max = mX_const_Max
    mX_Increment = mX_const_Increment
    mX_Values = ""
    mY_Min = mY_const_Min

```

```

mY_Max = mY_const_Max
mY_Heading = "Y-Axis"
mY_Increment = mY_const_Increment
mY_GridLines = False
Call DrawGraph
End Sub

```

```

Private Sub UserControl_MouseMove(Button As Integer, Shift As Integer, X As Single, Y
As Single)
    RaiseEvent MouseMove(Button, Shift, X, Y)
    Call DoCustomCursorIcon
End Sub

```

```

Private Sub UserControl_ReadProperties(PropBag As PropertyBag)
    mHeading = PropBag.ReadProperty("Heading", "Heading")
    mFont_Size = PropBag.ReadProperty("FontSize", m_const_Font_Size)
    mPrediction_DisplayInfo = PropBag.ReadProperty("Prediction_DisplayInfo", False)
    mPrediction_Heading_Bottom = PropBag.ReadProperty("Prediction_Heading_Bottom", "")
    mPrediction_Heading_Top = PropBag.ReadProperty("Prediction_Heading_Top", "")
    mPrediction_LineColor = PropBag.ReadProperty("Prediction_LineColor", vbWindowText)
    mSpecial_LineColor = PropBag.ReadProperty("Special_LineColor", vbWindowText)
    mSpecial_Display = PropBag.ReadProperty("Special_Display", False) 'False
    mSpecial_Lowerbound = PropBag.ReadProperty("Special_Lowerbound", 0)
    mSpecial_Upperbound = PropBag.ReadProperty("Special_Upperbound", 0)
    mUseCustomMouseIcon = PropBag.ReadProperty("UseCustomMouseIcon", False)
    bDoPicture = False
    Set MouseIcon = PropBag.ReadProperty("MouseIcon", Nothing)
    bDoPicture = True
    mX_Heading = PropBag.ReadProperty("XAxis_Heading", "X-Axis")
    mX_Min = PropBag.ReadProperty("XAxis_Min", mX_const_Min)
    mX_Max = PropBag.ReadProperty("XAxis_Max", mX_const_Max)
    mX_Increment = PropBag.ReadProperty("XAxis_Increment", mX_const_Increment)
    mX_Values = PropBag.ReadProperty("XAxis_Values", "")
    mY_Heading = PropBag.ReadProperty("YAxis_Heading", "Y-Axis")
    mY_Min = PropBag.ReadProperty("YAxis_Min", mY_const_Min)
    mY_Max = PropBag.ReadProperty("YAxis_Max", mY_const_Max)
    mY_Increment = PropBag.ReadProperty("YAxis_Increment", mY_const_Increment)
    mY_GridLines = PropBag.ReadProperty("YAxis_GridLines", False)
    bPropertiesRead = True
    Call DrawGraph
End Sub

```

```

Private Sub UserControl_Resize()
    lblHeading.Left = (UserControl.ScaleWidth - lblHeading.Width) / 2
    lblHeadingX.Move (UserControl.ScaleWidth - lblHeadingX.Width) / 2,
UserControl.ScaleHeight - lblHeadingX.Height - 4
    lblHeadingY.Top = (UserControl.ScaleHeight - lblHeadingY.Height) / 2
    If bPropertiesRead Then Call DrawGraph(False)
End Sub

```

```

Private Sub UserControl_WriteProperties(PropBag As PropertyBag)
    Call PropBag.WriteProperty("Heading", mHeading, "Heading")
    Call PropBag.WriteProperty("FontSize", mFont_Size, m_const_Font_Size)
    Call PropBag.WriteProperty("Prediction_DisplayInfo", mPrediction_DisplayInfo, False)
    Call PropBag.WriteProperty("Prediction_Heading_Bottom", mPrediction_Heading_Bottom,
"")
    Call PropBag.WriteProperty("Prediction_Heading_Top", mPrediction_Heading_Top, "")
    Call PropBag.WriteProperty("Prediction_LineColor", mPrediction_LineColor,
vbWindowText)
    Call PropBag.WriteProperty("Special_LineColor", mSpecial_LineColor, vbWindowText)
    Call PropBag.WriteProperty("Special_Display", mSpecial_Display, False)
    Call PropBag.WriteProperty("Special_Lowerbound", mSpecial_Lowerbound, 0)
    Call PropBag.WriteProperty("Special_Upperbound", mSpecial_Upperbound, 0)
    Call PropBag.WriteProperty("UseCustomMouseIcon", mUseCustomMouseIcon, False)
    Call PropBag.WriteProperty("MouseIcon", MouseIcon, Nothing)
    Call PropBag.WriteProperty("XAxis_Heading", mX_Heading, "X-Axis")
    Call PropBag.WriteProperty("XAxis_Min", mX_Min, mX_const_Min)
    Call PropBag.WriteProperty("XAxis_Max", mX_Max, mX_const_Max)
    Call PropBag.WriteProperty("XAxis_Increment", mX_Increment, mX_const_Increment)
    Call PropBag.WriteProperty("XAxis_Values", mX_Values, "")
    Call PropBag.WriteProperty("YAxis_Heading", mY_Heading, "Y-Axis")
    Call PropBag.WriteProperty("YAxis_Min", mY_Min, mY_const_Min)
    Call PropBag.WriteProperty("YAxis_Max", mY_Max, mY_const_Max)
    Call PropBag.WriteProperty("YAxis_Increment", mY_Increment, mY_const_Increment)
    Call PropBag.WriteProperty("YAxis_GridLines", mY_GridLines, False)
End Sub

```

```

Public Property Get Heading() As String
    Heading = mHeading
End Property

Public Property Let Heading(ByVal NewValue As String)
    mHeading = NewValue
    PropertyChanged "Heading"
    lblHeading.Caption = NewValue
    Call UserControl_Resize
End Property

Public Property Get FontSize() As Long
    FontSize = mFont_Size
End Property

Public Property Let FontSize(ByVal NewValue As Long)
    mFont_Size = NewValue
    PropertyChanged "FontSize"
    Call DrawGraph
End Property

Public Property Get MouseIcon() As Picture
    Set MouseIcon = picPicture.Picture
End Property

Public Property Set MouseIcon(ByVal NewValue As Picture)
    Set picPicture.Picture = NewValue
    PropertyChanged "MouseIcon"
End Property

Public Property Get Prediction_DisplayInfo() As Boolean
    Prediction_DisplayInfo = mPrediction_DisplayInfo
End Property

Public Property Let Prediction_DisplayInfo(ByVal NewValue As Boolean)
    mPrediction_DisplayInfo = NewValue
    PropertyChanged "Prediction_DisplayInfo"
End Property

Public Property Get Prediction_Heading_Bottom() As String
    Prediction_Heading_Bottom = mPrediction_Heading_Bottom
End Property

Public Property Let Prediction_Heading_Bottom(ByVal NewValue As String)
    mPrediction_Heading_Bottom = NewValue
    PropertyChanged "Prediction_Heading_Bottom"
End Property

Public Property Get Prediction_Heading_Top() As String
    Prediction_Heading_Top = mPrediction_Heading_Top
End Property

Public Property Let Prediction_Heading_Top(ByVal NewValue As String)
    mPrediction_Heading_Top = NewValue
    PropertyChanged "Prediction_Heading_Top"
End Property

Public Property Get Prediction_LineColor() As OLE_COLOR
    Prediction_LineColor = mPrediction_LineColor
End Property

Public Property Let Prediction_LineColor(ByVal NewValue As OLE_COLOR)
    mPrediction_LineColor = NewValue
    PropertyChanged "Prediction_LineColor"
End Property

Public Property Get Special_LineColor() As OLE_COLOR
    Special_LineColor = mSpecial_LineColor
End Property

Public Property Let Special_LineColor(ByVal NewValue As OLE_COLOR)
    mSpecial_LineColor = NewValue
    PropertyChanged "Special_LineColor"
    Call DisplaySpecialLines
End Property

Public Property Get Special_Display() As Boolean

```

```

    Special_Display = mSpecial_Display
End Property

Public Property Let Special_Display(ByVal NewValue As Boolean)
    mSpecial_Display = NewValue
    PropertyChanged "Special_Display"
    Call DisplaySpecialLines
End Property

Public Property Get Special_Lowerbound() As Long
    Special_Lowerbound = mSpecial_Lowerbound
End Property

Public Property Let Special_Lowerbound(ByVal NewValue As Long)
    mSpecial_Lowerbound = NewValue
    PropertyChanged "Special_Lowerbound"
    Call DisplaySpecialLines
End Property

Public Property Get Special_Upperbound() As Long
    Special_Upperbound = mSpecial_Upperbound
End Property

Public Property Let Special_Upperbound(ByVal NewValue As Long)
    mSpecial_Upperbound = NewValue
    PropertyChanged "Special_Upperbound"
    Call DisplaySpecialLines
End Property

Public Property Get UseCustomMouseIcon() As Boolean
    UseCustomMouseIcon = mUseCustomMouseIcon
End Property

Public Property Let UseCustomMouseIcon(ByVal NewValue As Boolean)
    mUseCustomMouseIcon = NewValue
    PropertyChanged "UseCustomMouseIcon"
End Property

Public Property Get XAxis_Heading() As String
    XAxis_Heading = mX_Heading
End Property

Public Property Let XAxis_Heading(ByVal NewValue As String)
    mX_Heading = NewValue
    PropertyChanged "XAxis_Heading"
    lblHeadingX.Caption = NewValue
    Call UserControl_Resize
End Property

Public Property Get XAxis_Min() As Long
    XAxis_Min = mX_Min
End Property

Public Property Let XAxis_Min(ByVal NewValue As Long)
    mX_Min = NewValue
    PropertyChanged "XAxis_Min"
    Call DrawGraph
End Property

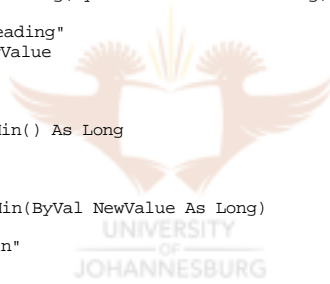
Public Property Get XAxis_Max() As Long
    XAxis_Max = mX_Max
End Property

Public Property Let XAxis_Max(ByVal NewValue As Long)
    mX_Max = NewValue
    PropertyChanged "XAxis_Max"
    Call DrawGraph
End Property

Public Property Get XAxis_Increment() As Long
    XAxis_Increment = mX_Increment
End Property

Public Property Let XAxis_Increment(ByVal NewValue As Long)
    mX_Increment = NewValue
    PropertyChanged "XAxis_Increment"
    Call DrawGraph

```



```

End Property

Public Property Get XAxis_Values() As String
    XAxis_Values = mX_Values
End Property

Public Property Let XAxis_Values(ByVal NewValue As String)
    mX_Values = NewValue
    PropertyChanged "XAxis_Values"
    Call UserControl_Resize
End Property

Public Property Get YAxis_Heading() As String
    YAxis_Heading = mY_Heading
End Property

Public Property Let YAxis_Heading(ByVal NewValue As String)
    mY_Heading = NewValue
    PropertyChanged "YAxis_Heading"
    lblHeadingY.Caption = NewValue
    Call UserControl_Resize
End Property

Public Property Get YAxis_Min() As Long
    YAxis_Min = mY_Min
End Property

Public Property Let YAxis_Min(ByVal NewValue As Long)
    mY_Min = NewValue
    PropertyChanged "YAxis_Min"
    Call DrawGraph
End Property

Public Property Get YAxis_Max() As Long
    YAxis_Max = mY_Max
End Property

Public Property Let YAxis_Max(ByVal NewValue As Long)
    mY_Max = NewValue
    PropertyChanged "YAxis_Max"
    Call DrawGraph
End Property

Public Property Get YAxis_Increment() As Long
    YAxis_Increment = mY_Increment
End Property

Public Property Let YAxis_Increment(ByVal NewValue As Long)
    mY_Increment = NewValue
    PropertyChanged "YAxis_Increment"
    Call DrawGraph
End Property

Public Property Get YAxis_GridLines() As Boolean
    YAxis_GridLines = mY_GridLines
End Property

Public Property Let YAxis_GridLines(ByVal NewValue As Boolean)
    mY_GridLines = NewValue
    PropertyChanged "YAxis_GridLines"
    Call DrawGraph
End Property

Private Sub DrawGraph(Optional bReloadObjects As Boolean = True)

    On Error GoTo ErrorHandler

    Dim lNumIncrements As Long, Count As Long
    Dim sngStart As Single, sngAddLeftTop As Single
    Dim lValues() As Long
    Dim bUseOwnValues As Boolean
    Dim XAxisValues() As String
    Dim lTemp1 As Long, lTemp2 As Long

    lblHeading.Caption = mHeading
    lblHeading.FontSize = mFont_Size
    lblHeadingX.Caption = mX_Heading

```

```

lblHeadingX.FontSize = mFont_Size
lblHeadingY.Caption = mY_Heading
lblHeadingY.FontSize = mFont_Size

linXAxis.X2 = UserControl.ScaleWidth - 14
If mPrediction_DisplayInfo Then
    linXAxis.Y1 = UserControl.ScaleHeight - 60
    linXAxis.Y2 = UserControl.ScaleHeight - 60
Else
    linXAxis.Y1 = UserControl.ScaleHeight - 45
    linXAxis.Y2 = UserControl.ScaleHeight - 45
End If
linYAxis.Y2 = UserControl.ScaleHeight - 40

If bReloadObjects Then
    If linX.Count > 1 Then
        For Count = 1 To linX.Count - 1
            Unload linX(Count)
            Unload lblXValue(Count)
        Next Count
    End If
End If

lNumIncrements = (mY_Max - mY_Min) / mY_Increment

bUseOwnValues = (mX_Values <> "")
If bUseOwnValues Then
    XAxisValues = Split(mX_Values, ",")

    If (UBound(XAxisValues) + 1) <> ((mX_Max - mX_Min) / mX_Increment) Then
        ReDim Preserve XAxisValues(0 To ((mX_Max - mX_Min) / mX_Increment) - 1)
    End If
End If

sngStart = linXAxis.Y1
sngAddLeftTop = (linXAxis.Y1 - linYAxis.Y1) / lNumIncrements
lblXValue(0).Caption = mY_Min
lblXValue(0).FontSize = mFont_Size - 2
lblXValue(0).AutoSize = False
lblXValue(0).Left = linXAxis.X1 - lblXValue(0).Width - 4
lblXValue(0).Top = sngStart - (lblXValue(0).Height / 2)
lblXValue(0).Visible = True
For Count = 1 To lNumIncrements
    If bReloadObjects Then
        Load linX(Count)
        Load lblXValue(Count)
    End If

    sngStart = sngStart - sngAddLeftTop
    linX(Count).X1 = linXAxis.X1
    If Not mY_GridLines Then
        linX(Count).X2 = linYAxis.X1
    End If
    linX(Count).Y1 = sngStart
    linX(Count).Y2 = sngStart
    linX(Count).Visible = True

    lblXValue(Count).AutoSize = True
    lblXValue(Count).FontSize = mFont_Size - 2
    lblXValue(Count).Caption = mY_Increment * Count
    lblXValue(Count).AutoSize = False
    lblXValue(Count).Left = linXAxis.X1 - lblXValue(Count).Width - 4
    lblXValue(Count).Top = sngStart - (lblXValue(0).Height / 2)
    lblXValue(Count).Visible = True
Next Count
linYAxis.Y1 = linX(lNumIncrements).Y1

If bReloadObjects Then
    If linY.Count > 1 Then
        For Count = 1 To linY.Count - 1
            Unload linY(Count)
            Unload lblYValue(Count)
        Next Count
    End If
End If

lNumIncrements = ((mX_Max - mX_Min) / mX_Increment)

```



```

If lNumIncrements <= 0 Then lNumIncrements = 1
sngStart = linYAxis.X1
sngAddLeftTop = (linXAxis.X2 - linXAxis.X1) / lNumIncrements
For Count = 1 To lNumIncrements
  If bReloadObjects Then
    Load linY(Count)
    Load lblYValue(Count)
  End If
  sngStart = sngStart + sngAddLeftTop
  linY(Count).Y1 = linXAxis.Y1
  linY(Count).Y2 = linY(Count).Y1 + 5
  linY(Count).X1 = sngStart
  linY(Count).X2 = sngStart
  linY(Count).Visible = True

  lblYValue(Count).AutoSize = True
  lblYValue(Count).FontSize = mFont_Size - 2
  If bUseOwnValues Then
    lblYValue(Count).Caption = XAxisValues(Count - 1)
  Else
    lblYValue(Count).Caption = mX_Increment * Count
  End If
  lblYValue(Count).AutoSize = False
  If Count = 1 Then
    lblYValue(Count).Left = ((sngStart + linYAxis.X1) / 2) - (lblYValue(Count).Width
/ 2)
  Else
    lblYValue(Count).Left = ((sngStart + linY(Count - 1).X1) / 2) -
(lblYValue(Count).Width / 2)
  End If
  lblYValue(Count).Top = linXAxis.Y1 + 6
  lblYValue(Count).Visible = True
Next Count
linXAxis.X2 = linY(lNumIncrements).X1

If mY_GridLines Then
  For Count = 1 To linX.Count - 1
    linX(Count).X2 = linXAxis.X2
  Next Count
  linY(linX.Count - 1).Y1 = linX(linX.Count - 1).Y1
End If

If shpBar.Count > 1 Then
  ReDim lValues(1 To lblColumnVal.Count - 1)

  For Count = 1 To lblColumnVal.Count - 1
    lValues(Count) = CLng(lblColumnVal(Count).Caption)
  Next Count

  If Not bReloadObjects Then bColumnBarsAlreadyLoaded = True
  Call DrawGraphColumns(lValues)
End If

Call DisplaySpecialLines

Quit:
Exit Sub

ErrorHandler:
If Err = 380 Then
  mFont_Size = 10
  Resume
End If
MsgBox Err.Description, vbCritical, "Error" + Str(Err)
Resume Quit
Resume

End Sub

Public Sub DrawGraphColumns(ColumnValues() As Long)

  On Error GoTo ErrorHandler

  Dim lNumColumns As Long, Count As Long
  Dim sngWidth As Single, sngTop As Single
  Dim Val As Long
  Dim nIndex As Integer

```



```

Dim Percentage As Double

'Unload previously loaded column bars
If Not bColumnBarsAlreadyLoaded Then
  If shpBar.Count > 1 Then
    For Count = 1 To shpBar.Count - 1
      Unload shpBar(Count)
      Unload lblColumnVal(Count)
    Next Count
  End If
End If

lNumColumns = UBound(ColumnValues)
If lNumColumns > ((mX_Max - mX_Min) / mX_Increment) Then lNumColumns = ((mX_Max -
mX_Min) / mX_Increment)
If lNumColumns < 0 Then Exit Sub

linX(0).Y1 = linXAxis.Y1
linX(0).Y2 = linXAxis.Y2

sngWidth = (linY(1).X1 - linYAxis.X1) / 2
For Count = 1 To lNumColumns
  If Not bColumnBarsAlreadyLoaded Then
    Load shpBar(Count)
    Load lblColumnVal(Count)
  End If

  Val = ColumnValues(Count)
  lblColumnVal(Count).AutoSize = True
  lblColumnVal(Count).FontSize = mFont_Size - 2
  lblColumnVal(Count).Caption = Val
  lblColumnVal(Count).AutoSize = False

  shpBar(Count).Width = sngWidth
  If Count = 1 Then
    shpBar(Count).Left = ((linY(1).X1 + linYAxis.X1) / 2) - (sngWidth / 2)
    lblColumnVal(Count).Left = ((linY(1).X1 + linYAxis.X1) / 2) -
(lblColumnVal(Count).Width / 2)
  Else
    shpBar(Count).Left = ((linY(Count).X1 + linY(Count - 1).X1) / 2) - (sngWidth /
2)
    lblColumnVal(Count).Left = ((linY(Count).X1 + linY(Count - 1).X1) / 2) -
(lblColumnVal(Count).Width / 2)
  End If

  If Val > 0 Then
    nIndex = 0
    Do While Val > CLng(lblXValue(nIndex).Caption)
      nIndex = nIndex + 1
    Loop
  AfterTooManyElements:
    Percentage = (Val - CLng(lblXValue(nIndex - 1).Caption)) /
(CLng(lblXValue(nIndex).Caption) - CLng(lblXValue(nIndex - 1).Caption))
    sngTop = (linX(nIndex - 1).Y1) - ((linX(nIndex - 1).Y1 - linX(nIndex).Y1) *
Percentage)
    shpBar(Count).Top = sngTop

    shpBar(Count).Height = linXAxis.Y1 - sngTop

    lblColumnVal(Count).Top = sngTop - (lblColumnVal(Count).Height * 1.5)

    shpBar(Count).Visible = True
  Else
    lblColumnVal(Count).Top = linXAxis.Y1 - (lblColumnVal(Count).Height * 1.25)
    shpBar(Count).Visible = False
  End If

  lblColumnVal(Count).Visible = True

Next Count

Quit:
bColumnBarsAlreadyLoaded = False
Exit Sub

ErrorHandler:
If Err = 340 Then

```

```

    nIndex = nIndex - 1
    Resume AfterTooManyElements
End If
Resume Quit
Resume

End Sub

```

```

Public Function GetGraphColumnValues(ColumnValues() As Long) As Long

    On Error Resume Next

    Dim lNumCols As Long
    Dim Count As Long

    lNumCols = (lblColumnVal.Count - 1)
    If lNumCols <= 0 Then
        lNumCols = 0
        ReDim ColumnValues(lNumCols)
    Else
        ReDim ColumnValues(1 To lNumCols)
    End If

    If lNumCols > 0 Then
        For Count = 1 To (lblColumnVal.Count - 1)
            ColumnValues(Count) = CLng(lblColumnVal(Count).Caption)
        Next Count
    End If

    GetGraphColumnValues = lNumCols

End Function

```

```

Private Sub DoCustomCursorIcon()

    Dim Count As Long

    If mUseCustomMouseIcon = True Then
        If picPicture.Picture <> LoadPicture("") Then
            UserControl.MousePointer = vbCustom
            Set UserControl.MouseIcon = picPicture.Picture
            lblHeading.MousePointer = vbCustom
            Set lblHeading.MouseIcon = picPicture.Picture
            lblHeadingX.MousePointer = vbCustom
            Set lblHeadingX.MouseIcon = picPicture.Picture

            For Count = 1 To (lblXValue.Count - 1)
                lblXValue(Count).MousePointer = vbCustom
                Set lblXValue(Count).MouseIcon = picPicture.Picture
            Next Count
            For Count = 1 To (lblYValue.Count - 1)
                lblYValue(Count).MousePointer = vbCustom
                Set lblYValue(Count).MouseIcon = picPicture.Picture
            Next Count
            For Count = 1 To (lblColumnVal.Count - 1)
                lblColumnVal(Count).MousePointer = vbCustom
                Set lblColumnVal(Count).MouseIcon = picPicture.Picture
            Next Count
        Else
            UserControl.MousePointer = vbNormal
            lblHeading.MousePointer = vbNormal
            lblHeadingX.MousePointer = vbNormal

            For Count = 1 To (lblXValue.Count - 1)
                lblXValue(Count).MousePointer = vbNormal
            Next Count
            For Count = 1 To (lblYValue.Count - 1)
                lblYValue(Count).MousePointer = vbNormal
            Next Count
            For Count = 1 To (lblColumnVal.Count - 1)
                lblColumnVal(Count).MousePointer = vbNormal
            Next Count
        End If
    Else
        UserControl.MousePointer = vbNormal
        lblHeading.MousePointer = vbNormal
        lblHeadingX.MousePointer = vbNormal
    End If

```

```

For Count = 1 To (lblXValue.Count - 1)
    lblXValue(Count).MousePointer = vbNormal
Next Count
For Count = 1 To (lblYValue.Count - 1)
    lblYValue(Count).MousePointer = vbNormal
Next Count
For Count = 1 To (lblColumnVal.Count - 1)
    lblColumnVal(Count).MousePointer = vbNormal
Next Count
End If
End Sub

Private Sub DisplaySpecialLines()

Dim sngTop As Single

If mSpecial_Display Then
    linSpecialTop.BorderColor = mSpecial_LineColor
    linSpecialBottom.BorderColor = mSpecial_LineColor

    'Get top line values
    sngTop = GetLineYValues(mSpecial_Upperbound)
    linSpecialTop.X1 = linYAxis.X1 + 1
    If mY_GridLines Then
        linSpecialTop.X2 = linXAxis.X2
    Else
        linSpecialTop.X2 = linXAxis.X2
    End If
    linSpecialTop.Y1 = sngTop
    linSpecialTop.Y2 = sngTop

    'Get bottom line values
    sngTop = GetLineYValues(mSpecial_Lowerbound)
    linSpecialBottom.X1 = linYAxis.X1 + 1
    If mY_GridLines Then
        linSpecialBottom.X2 = linXAxis.X2
    Else
        linSpecialBottom.X2 = linXAxis.X2
    End If
    linSpecialBottom.Y1 = sngTop
    linSpecialBottom.Y2 = sngTop

    linSpecialTop.Visible = True
    linSpecialBottom.Visible = True
Else
    linSpecialTop.Visible = False
    linSpecialBottom.Visible = False
End If

End Sub

Public Sub DrawGraphWithPredictions(ColumnValues() As Long, PredictionLowerBounds() As Long, PredictionUpperBounds() As Long)

On Error GoTo ErrorHandler

Dim lNumColumns As Long, Count As Long
Dim sngWidth As Single, sngTop As Single, sngLeft As Single
Dim Val As Long
Dim nIndex As Integer
Dim Percentage As Double

If (UBound(ColumnValues) < 1) Or (UBound(PredictionLowerBounds) < 1) Or (UBound(PredictionUpperBounds) < 1) Then Exit Sub

Call DrawGraphColumns(ColumnValues)

'Unload previously loaded bound-lines
For Count = 1 To (linPredictTop.Count - 1)
    Unload shpPredict(Count)
    Unload linPredictTop(Count)
'Unload linPredictBottom(Count)
Next Count

If linY.Count > 1 Then

```

```

For Count = 1 To (linY.Count - 1)
  Load linPredictTop(Count)
  Load shpPredict(Count)

  If PredictionLowerBounds(Count) = PredictionUpperBounds(Count) Then
    linPredictTop(Count).BorderColor = mPrediction_LineColor
    If Count = 1 Then
      linPredictTop(Count).X1 = linYAxis.X1 + 1
    Else
      linPredictTop(Count).X1 = linY(Count - 1).X1 + 1
    End If
    linPredictTop(Count).X2 = linY(Count).X1
    sngTop = GetLineYValues(PredictionUpperBounds(Count))
    linPredictTop(Count).Y1 = sngTop
    linPredictTop(Count).Y2 = sngTop
    linPredictTop(Count).Visible = True
    linPredictTop(Count).ZOrder 0
    'If sngTop <> linXAxis.Y1 Then linPredictTop(Count).ZOrder 0
  Else
    shpPredict(Count).BorderColor = mPrediction_LineColor
    shpPredict(Count).FillColor = mPrediction_LineColor
    shpPredict(Count).FillStyle = vbSolid
    If Count = 1 Then
      shpPredict(Count).Left = linYAxis.X1 + 1
      shpPredict(Count).Width = linY(Count).X1 - (linYAxis.X1 + 1)
    Else
      shpPredict(Count).Left = linY(Count - 1).X1 + 1
      shpPredict(Count).Width = linY(Count).X1 - (linY(Count - 1).X1 + 1)
    End If
    sngTop = GetLineYValues(PredictionUpperBounds(Count))
    shpPredict(Count).Top = sngTop
    shpPredict(Count).Height = GetLineYValues(PredictionLowerBounds(Count)) -
sngTop
    shpPredict(Count).Visible = True
    shpPredict(Count).ZOrder 1
  End If

  'Load linPredictBottom(Count)
  'linPredictBottom(Count).BorderColor = mPrediction_LineColor
  'linPredictBottom(Count).X1 = linPredictTop(Count).X1
  'linPredictBottom(Count).X2 = linPredictTop(Count).X2
  'sngTop = GetLineYValues(PredictionLowerBounds(Count))
  'linPredictBottom(Count).Y1 = sngTop
  'linPredictBottom(Count).Y2 = sngTop
  'linPredictBottom(Count).Visible = True
  'linPredictBottom(Count).ZOrder 0
Next Count
End If

If mPrediction_DisplayInfo Then
  lblInfo(0).Caption = mPrediction_Heading_Top
  lblInfo(1).Caption = mPrediction_Heading_Bottom
  If TextWidth(mPrediction_Heading_Top) > TextWidth(mPrediction_Heading_Bottom) Then
    sngLeft = linXAxis.X2 - lblInfo(0).Width
  Else
    sngLeft = linXAxis.X2 - lblInfo(1).Width
  End If
  lblInfo(0).Left = sngLeft
  lblInfo(1).Left = sngLeft
  lblInfo(0).Top = lblYValue(lblYValue.Count - 1).Top + lblYValue(lblYValue.Count -
1).Height + 5
  lblInfo(1).Top = lblInfo(0).Top + lblInfo(0).Height - 3
  shpInfo.Left = sngLeft - shpInfo.Width - 10
  shpInfo.Top = lblInfo(1).Top + ((lblInfo(1).Height - shpInfo.Height) / 2)
  linInfo.BorderColor = mPrediction_LineColor
  linInfo.Y1 = lblInfo(0).Top + (lblInfo(0).Height / 2)
  linInfo.Y2 = linInfo.Y1
  linInfo.X1 = shpInfo.Left
  linInfo.X2 = shpInfo.Left + shpInfo.Width

  lblInfo(0).Visible = True
  lblInfo(1).Visible = True
  linInfo.Visible = True
  shpInfo.Visible = True
Else
  lblInfo(0).Visible = False
  lblInfo(1).Visible = False

```

```

        linInfo.Visible = False
        shpInfo.Visible = False
    End If

Quit:
    Exit Sub

ErrorHandler:
    Resume Quit
Resume

End Sub

```

```

Private Function GetLineYValues(ColumnValue As Long) As Single

    Dim Val As Long
    Dim nIndex As Integer
    Dim Percentage As Double
    Dim sngTop As Single

    If ColumnValue = 0 Then
        GetLineYValues = linXAxis.Y1 - 1
        Exit Function
    End If

    nIndex = 0
    Val = ColumnValue
    Do While Val > CLng(lblXValue(nIndex).Caption)
        nIndex = nIndex + 1
    Loop
    Percentage = (Val - CLng(lblXValue(nIndex - 1).Caption)) /
    (CLng(lblXValue(nIndex).Caption) - CLng(lblXValue(nIndex - 1).Caption))
    sngTop = (linX(nIndex - 1).Y1) - ((linX(nIndex - 1).Y1 - linX(nIndex).Y1) *
    Percentage)
    GetLineYValues = sngTop

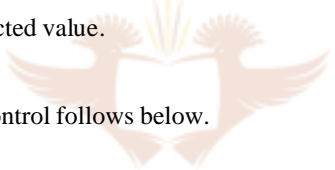
End Function

```

B.3.2 The “SumView” control

The “SumView” control is used to draw a summation function in a step when calculating the fuzzy expected value.

The source code for this control follows below.



```

Option Explicit

'Constants
Private Const m_const_Variable As String = "i"
Private Const m_const_FromValue As String = "1"
Private Const m_const_ToValue As String = "1"

'Values
Private mVariable As String
Private mFromValue As String
Private mToValue As String

```

```

Private Sub UserControl_InitProperties()
    mVariable = m_const_Variable
    mFromValue = m_const_FromValue
    mToValue = m_const_ToValue
End Sub

```

```

Private Sub UserControl_ReadProperties(PropBag As PropertyBag)
    mVariable = PropBag.ReadProperty("Variable", m_const_Variable)
    mFromValue = PropBag.ReadProperty("FromValue", m_const_FromValue)
    mToValue = PropBag.ReadProperty("ToValue", m_const_ToValue)
    Call DrawSum

```

```

End Sub
-----
Private Sub UserControl_Resize()

    If lblFrom.Width > imgSum.Width Then
        UserControl.Width = (lblFrom.Width + 20) \ Screen.TwipsPerPixelX
    Else
        UserControl.Width = (imgSum.Width + 20) \ Screen.TwipsPerPixelX
    End If
    UserControl.Height = (lblFrom.Top + lblFrom.Height + 20) * Screen.TwipsPerPixelY

End Sub
-----
Private Sub UserControl_WriteProperties(PropBag As PropertyBag)
    Call PropBag.WriteProperty("Variable", mVariable, m_const_Variable)
    Call PropBag.WriteProperty("FromValue", mFromValue, m_const_FromValue)
    Call PropBag.WriteProperty("ToValue", mToValue, m_const_ToValue)
End Sub
-----
Public Property Get Variable() As String
    Variable = mVariable
End Property
-----
Public Property Let Variable(ByVal NewValue As String)
    mVariable = NewValue
    PropertyChanged "Variable"
    Call DrawSum
End Property
-----
Public Property Get FromValue() As String
    FromValue = mFromValue
End Property
-----
Public Property Let FromValue(ByVal NewValue As String)
    mFromValue = NewValue
    PropertyChanged "FromValue"
    Call DrawSum
End Property
-----
Public Property Get ToValue() As String
    ToValue = mToValue
End Property
-----
Public Property Let ToValue(ByVal NewValue As String)
    mToValue = NewValue
    PropertyChanged "ToValue"
    Call DrawSum
End Property
-----
Private Sub DrawSum()

    lblFrom.Caption = mVariable & "=" & mFromValue
    lblTo.Caption = mToValue

    lblTo.Top = 0
    imgSum.Top = lblTo.Height + 10 \ (5 * Screen.TwipsPerPixelY)
    lblFrom.Top = imgSum.Top + imgSum.Height + 10 \ (10 * Screen.TwipsPerPixelY)

    If lblFrom.Width > imgSum.Width Then
        lblFrom.Left = 0
        imgSum.Left = (lblFrom.Width - imgSum.Width) / 2
        lblTo.Left = (lblFrom.Width - lblTo.Width) / 2
    Else
        imgSum.Left = 0
        lblFrom.Left = (imgSum.Width - lblFrom.Width) / 2
        lblTo.Left = (imgSum.Width - lblTo.Width) / 2
    End If
    Call UserControl_Resize

End Sub
-----

```

B.3.3 The “VertLabel” control

The “VertLabel” control is used to draw the bars on a graph.

The source code for this control follows below.

```
Option Explicit

Dim bPropertiesRead As Boolean

Const m_const_Font_Size As Long = 10

Private mCaption As String
Private mFont_Size As String
Private mBackColor As OLE_COLOR

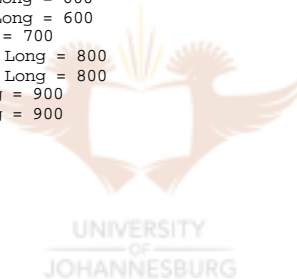
'API functions
Private Declare Function CreateFontIndirect Lib "gdi32" Alias "CreateFontIndirectA"
(lpLogFont As LOGFONT) As Long
Private Declare Function SelectObject Lib "gdi32" (ByVal hdc As Long, ByVal hObject As
Long) As Long
Private Declare Function DeleteObject Lib "gdi32" (ByVal hObject As Long) As Long
Private Declare Function GetTextExtentPoint32 Lib "gdi32" Alias
"GetTextExtentPoint32A" (ByVal hdc As Long, ByVal lpstr As String, ByVal cbString As
Long, lpSize As POINTAPI) As Long

'Constants
Private Const LF_FACESIZE As Long = 32
Private Const FW_DONTCARE As Long = 0
Private Const FW_THIN As Long = 100
Private Const FW_EXTRALIGHT As Long = 200
Private Const FW_ULTRALIGHT As Long = 200
Private Const FW_LIGHT As Long = 300
Private Const FW_NORMAL As Long = 400
Private Const FW_REGULAR As Long = 400
Private Const FW_MEDIUM As Long = 500
Private Const FW_SEMIBOLD As Long = 600
Private Const FW_DEMIBOLD As Long = 600
Private Const FW_BOLD As Long = 700
Private Const FW_EXTRABOLD As Long = 800
Private Const FW_ULTRABOLD As Long = 800
Private Const FW_HEAVY As Long = 900
Private Const FW_BLACK As Long = 900

'Types
Private Type LOGFONT
    lfHeight As Long
    lfWidth As Long
    lfEscapement As Long
    lfOrientation As Long
    lfWeight As Long
    lfItalic As Byte
    lfUnderline As Byte
    lfStrikeOut As Byte
    lfCharSet As Byte
    lfOutPrecision As Byte
    lfClipPrecision As Byte
    lfQuality As Byte
    lfPitchAndFamily As Byte
    lfFaceName As String * LF_FACESIZE
End Type

Private Type POINTAPI
    X As Long
    Y As Long
End Type

Public Event Click()
Public Event MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
```



```

Private Sub UserControl_Click()
    RaiseEvent Click
End Sub

Private Sub UserControl_InitProperties()
    mCaption = "Caption"
    mFont_Size = m_const_Font_Size
    mBackColor = vbButtonFace
    Call DrawLabel
End Sub

Private Sub UserControl_MouseMove(Button As Integer, Shift As Integer, X As Single, Y
As Single)
    RaiseEvent MouseMove(Button, Shift, X, Y)
End Sub

Private Sub UserControl_Paint()
    Call DrawLabel
End Sub

Private Sub UserControl_ReadProperties(PropBag As PropertyBag)
    mCaption = PropBag.ReadProperty("Caption", "Caption")
    mBackColor = PropBag.ReadProperty("BackColor", vbButtonFace)
    mFont_Size = PropBag.ReadProperty("FontSize", m_const_Font_Size)
    bPropertiesRead = True
    Call DrawLabel
End Sub

Private Sub UserControl_Resize()
    If bPropertiesRead Then Call DrawLabel
End Sub

Private Sub UserControl_WriteProperties(PropBag As PropertyBag)
    Call PropBag.WriteProperty("Caption", mCaption, "Caption")
    Call PropBag.WriteProperty("BackColor", mBackColor, vbButtonFace)
    Call PropBag.WriteProperty("FontSize", mFont_Size, m_const_Font_Size)
End Sub

Public Property Get Caption() As String
    Caption = mCaption
End Property

Public Property Let Caption(ByVal NewValue As String)
    mCaption = NewValue
    PropertyChanged "Caption"
    Call DrawLabel
End Property

Public Property Get BackColor() As OLE_COLOR
    BackColor = mBackColor
End Property

Public Property Let BackColor(ByVal NewValue As OLE_COLOR)
    mBackColor = NewValue
    PropertyChanged "BackColor"
    Call DrawLabel
End Property

Public Property Get FontSize() As Long
    FontSize = mFont_Size
End Property

Public Property Let FontSize(ByVal NewValue As Long)
    mFont_Size = NewValue
    PropertyChanged "FontSize"
    Call DrawLabel
End Property

Private Sub DrawLabel()

    Dim font As LOGFONT
    Dim prevFont As Long, hFont As Long, ret As Long
    Dim TextSize As POINTAPI

    UserControl.Cls
    UserControl.BackColor = mBackColor

```



```
font.lfEscapement = 900
font.lfFaceName = "Arial" & Chr$(0) 'Null character at end, ONLY True-type fonts
font.lfWeight = FW_BOLD

'Windows expects the font size to be in pixels and to
'be negative if you are specifying the character height
'you want.
If mFont_Size = "" Then
    mFont_Size = "0"
End If
font.lfHeight = (mFont_Size * -20) / Screen.TwipsPerPixelY
hFont = CreateFontIndirect(font)
prevFont = SelectObject(UserControl.hdc, hFont)

'Get the height and width of our text
GetTextExtentPoint32 UserControl.hdc, mCaption, Len(mCaption), TextSize

UserControl.CurrentX = 0 'usercontrol.ScaleWidth / 2
UserControl.CurrentY = (TextSize.X * Screen.TwipsPerPixelY) 'usercontrol.ScaleHeight
/ 2
DoEvents
UserControl.Print mCaption "Rotated Text"
' Clean up by restoring original font.
ret = SelectObject(UserControl.hdc, prevFont)
ret = DeleteObject(hFont)

UserControl.Width = (TextSize.Y * Screen.TwipsPerPixelX)
UserControl.Height = (TextSize.X * Screen.TwipsPerPixelY)

End Sub
```





APPENDIX C

CYBERCOP SCANNER REPORT

The CyberCop Scanner report that was created for the specific scan scenario as stated in Chapter 8. The report was 162 pages long. An extract of that report for one specific host, **eclab173.rau.ac.za**, is shown in this appendix.

CyberCop Scanner Results

Report Sorted By Host



152.106.42.173
eclab173.rau.ac.za

14 Vulnerabilities

1041 Trace route to host

Risk Factor: Low
Complexity: Low
Popularity: Popular
Impact: Intelligence
Root Cause: Insecure Design
Ease of Fix: Moderate
Description: This module traces the route to the host being scanned in the same manner as the traceroute program in UNIX or the tracert program in Windows NT. The route information is stored to the network map file as well as being returned by the module. The network mapper uses this information to build a map of the network.

Security Concerns: By allowing traceroutes into your network from outside you allow detailed network maps to be derived from the information available. Targets for exploitation can be determined from these maps. This presents a strong enticement risk.

Suggestion: Block all unnecessary ICMP, UDP and TCP ports, and loose and strict source routed packets. This is usually accomplished with firewall and network routing technology. Protect your sensitive servers with such technology where possible.

16020 NetBIOS Name Table Retrieval**Risk Factor:** Low**Complexity:** Medium**Popularity:** Widespread**Impact:** Intelligence**Root Cause:** Misconfiguration**Ease of Fix:** Moderate**Description:** This check obtains the system name tables from the remote system's NetBIOS name service.**Security Concerns:** By accessing system name table information, individuals can obtain information which can be used to launch an attack. Information available includes:

1. The NetBIOS name of the server.
2. The Windows NT workgroup domain name.
3. Login names of users who are logged into the server.
4. The name of the administrator account if they are logged into the server.

Suggestion: Ensure that users outside of your network are not permitted to access the NetBIOS name service. This can be performed by implementing packet filters on UDP port 137.**18001 Connection to IPC\$ as Anonymous User Allowed****Risk Factor:** Low**Complexity:** Medium**Popularity:** Widespread**Impact:** Intelligence**Root Cause:** Misconfiguration**Ease of Fix:** Simple**Description:** The remote host allows the Anonymous user to establish connections to the IPC\$ share over the network. The IPC\$ share is used by Windows NT to provide a number of system administration services to other networked users.

Unix machines running the Samba SMB service also make an IPC\$ share available over the network.

Security Concerns: By default, various services and pipes are offered by the IPC\$ share which cannot be easily restricted by Windows NT.**Suggestion:** It is suggested that you ensure proper restrictions are present to disallow connections to IPC\$ from entering your network. This can be performed by disallowing TCP port 139 from being accessed by the outside network. Ensure that you are aware that restricting access to port 139 may limit the functionality of Windows NT to the outside network. This should be performed by preventing your firewall or router from passing TCP port 139. Consult the Samba documentation for more information about this issue under Unix.

18024 **Unable to access IPC\$ or Registry****Risk Factor:** Low**Complexity:** Low**Popularity:** Widespread**Impact:** System Integrity::Authorization ::Intelligence**Root Cause:** Misconfiguration**Ease of Fix:** Simple

Description: CyberCop Scanner was unable to obtain full access to the target host's IPC\$ share, or the Windows NT registry. Many of the policy checks in the scanner require access to the IPC\$ share or to the registry of the machine being scanned. Without the proper access, some checks will not be able to detect vulnerabilities on the remote machine. This module provides a warning specifying when access to the IPC\$ share, the HKEY_LOCAL_MACHINE registry hive or the HKEY_USERS registry hive was not granted. This indicates that a complete audit of the target system may not have been performed.

This can occur if the account the scan is being run from does not have access to the machine being scanned or if the account does not have sufficient permission to access the remote resources.

This may also indicate that the machine is a standalone system, or is not part of the same Windows NT domain from which the scan is being performed.

If access to the registry was not obtained, it may also indicate that the target system is not a Windows NT system.

Security Concerns:

Suggestion: Ensure that you have run the scanner as the domain Administrator, who has sufficient access to perform auditing of the target system.

21001 **TCP port scanning****Risk Factor:** Low**Complexity:** Low**Popularity:** Popular**Impact:** Intelligence**Root Cause:** Insecure Design**Ease of Fix:** Difficult**Description:** This check scans a target host for listening TCP ports.**Security Concerns:**

Suggestion: The scanner will return which TCP ports are listening. You should check these ports to see if they are running services that you have approved. If they are running services which are undocumented, or which you do not wish to run, we suggest you disable them.

Many operating systems are shipped with a large number of services that are not required for normal operation. In some cases these services may contain known or unknown security problems. It is recommended that any services which are not required be disabled.

21002 UDP scanning check

Risk Factor: Low
Complexity: Medium
Popularity: Popular
Impact: Intelligence
Root Cause: Insecure Design
Ease of Fix: Difficult

Description: This check scans a target host for listening UDP ports. Scanning for active UDP ports is very difficult to perform reliably. This is due to the fact that UDP is a connectionless protocol, and there is no reliable indication whether or not a connection has been established. There are 2 primary methods used to scan for listening UDP ports:

1. Sending data to a UDP port, and awaiting a response from that port.
2. Sending data to a UDP port, and awaiting an ICMP port unreachable message, indicating that this port is NOT active. This allows us to build a listing of ports which may be active (if no port unreachable message is received from that port).

There are problems when using both methods. When using method 1 and sending random data to each UDP port, many services will not respond if they cannot recognize the data. This results in being unable to detect many UDP servers which may be running. Using method 2 is reliable if we can ensure that two conditions are met:

1. No ICMP port unreachable messages are lost in transit.
2. The host reliably returns an ICMP port unreachable packet for every port that is inactive. This varies from operating system to operating system, in that certain operating systems implement thresholds to prevent themselves from sending out too many ICMP port unreachable messages in a period of time. Examples of this threshold have been found in versions of Linux and Solaris.

CyberCop Scanner attempts to determine the best method for scanning a host for listening UDP servers. It's first choice is to scan by sending data and watching for ICMP unreachable messages. CyberCop Scanner will determine whether this is possible by first attempting this on ports 45000-45009. If CyberCop Scanner receives back all 10 ICMP port unreachable messages, it will use this method to scan for active UDP services, and assumes that the host reliably returns ICMP port unreachable messages. If this test fails, then method 1 is used, and data is sent to each port, awaiting a response. If method 2 was used, CyberCop Scanner will attempt to verify results by sending 2 more sets of data packets, and ensuring that the host is not returning ICMP port unreachable messages for ports which were found to be active earlier. This is an attempt to ensure that if any ICMP port unreachable packets were lost in transit, we do not falsely report listening ports.

The results from this scan are fairly reliable when scanning on the local network, however will vary on long haul networks. Filtering routers will also cause results to vary.

Note that this module can cause inferior routing software to fail. This module safely evaluates all major network operating systems.

Security Concerns:

Suggestion: The scanner will return which UDP ports are listening. You should check these ports to see if they are running services that you have approved. If they are running services which are undocumented, or which you do not wish to run, we suggest you disable them. Many operating systems are shipped with a large number of services that are not required for normal operation. In some cases these services may contain known or unknown security problems. It is recommended that any services which are not required be disabled.

21003 **TCP SYN port scanning**

Risk Factor: Low

Complexity: Medium

Popularity: Popular

Impact: Intelligence

Root Cause: Insecure Design

Ease of Fix: Difficult

Description: This check can be used as a much faster alternative to regular TCP port scanning. This check scans a target host for listening TCP ports in much the same way as the regular TCP port scanning, however does so by sending a packet to initiate a connection and watching for a response. The difference in using this method is that a complete connection to the remote host is not actually opened. The drawback in using this method is that it may be unreliable due to packet loss on the network.

Security Concerns:

Suggestion: The scanner will return which TCP ports are listening. You should check these ports to see if they are running services that you have approved. If they are running services which are undocumented, or which you do not wish to run, we suggest you disable them. Many operating systems are shipped with a large number of services that are not required for normal operation. In some cases these services may contain known or unknown security problems. It is recommended that any services which are not required be disabled.



26001 User Enumeration via Anonymous Logon

Risk Factor: Low
Complexity: Low
Popularity: Popular
Impact: Intelligence
Root Cause: Software Implementation Problems
Ease of Fix: Trivial
Description: A listing of user accounts present on the target host was retrieved. Windows NT provides enumeration functions for enumerating users on the network. By default, Windows NT 4.0 and 3.51 allow anonymous logon users (also known as NULL session connections) to list account names.

Security Concerns:

Suggestion: To prevent the ability for Anonymous users to enumerate users, create the following registry key:
Hive : HKEY_LOCAL_MACHINE
Key : System\CurrentControlSet\Control\LSA
Name : RestrictAnonymous
Type : REG_DWORD
Value: 1
Please note that Service Pack 3 must be installed for these restrictions to function.

26002 Active Users Enumeration via Anonymous Logon

Risk Factor: Low
Complexity: Low
Popularity: Popular
Impact: Intelligence
Root Cause: Software Implementation Problems
Ease of Fix: Trivial
Description: A listing of logged in users on the target host was retrieved. Windows NT provides enumeration functions for enumerating users on the network. By default, Windows NT 4.0 and 3.51 allow anonymous logon users (also known as NULL session connections) to list account names.

Security Concerns:

Suggestion: To prevent the ability for Anonymous users to enumerate users, create the following registry key:
Hive : HKEY_LOCAL_MACHINE
Key : System\CurrentControlSet\Control\LSA
Name : RestrictAnonymous
Type : REG_DWORD
Value: 1
Please note that Service Pack 3 must be installed for these restrictions to function.

26003 Group Enumeration via Anonymous Logon

Risk Factor: Low
Complexity: Low
Popularity: Popular
Impact: Intelligence
Root Cause: Software Implementation Problems
Ease of Fix: Trivial

Description: A listing of groups present on the target host was retrieved. Windows NT provides enumeration functions for enumerating groups on the network. By default, Windows NT 4.0 and 3.51 allow anonymous logon users (also known as NULL session connections) to list group names.

Security Concerns:

Suggestion: To prevent the ability for Anonymous users to enumerate groups, create the following registry key:
Hive : HKEY_LOCAL_MACHINE
Key : System\CurrentControlSet\Control\LSA
Name : RestrictAnonymous
Type : REG_DWORD
Value: 1
Please note that Service Pack 3 must be installed for these restrictions to function.

26004 Share Enumeration via Anonymous Logon

Risk Factor: Low
Complexity: Low
Popularity: Popular
Impact: Intelligence
Root Cause: Misconfiguration
Ease of Fix: Trivial

Description: A listing of shares present on the target host was retrieved. Windows NT provides enumeration functions for enumerating shares on the network. By default, Windows NT 4.0 and 3.51 allow anonymous logon users (also known as NULL session connections) to list shares.

Security Concerns:

Suggestion: To prevent the ability for Anonymous users to enumerate shares, create the following registry key:
Hive : HKEY_LOCAL_MACHINE
Key : System\CurrentControlSet\Control\LSA
Name : RestrictAnonymous
Type : REG_DWORD
Value: 1
Please note that Service Pack 3 must be installed for these restrictions to function.

26005 Enumerate Network Transports via Anonymous Logon

Risk Factor: Low
Complexity: Low
Popularity: Popular
Impact: Intelligence
Root Cause: Software Implementation Problems
Ease of Fix: Trivial
Description: CyberCop Scanner was able to retrieve a listing of network transports which are present on the target host. Windows NT provides functions for enumerating the transports on a network. This module uses these functions to enumerate all the network transports on a machine. This provides a list of the networking transports installed on a machine as well as the hardware addresses of the network cards bound to the transports.

Security Concerns:

Suggestion: There is currently no method to disable the enumeration of network transports via the Anonymous user account.

26006 Enumerate Active Sessions via Anonymous Logon

Risk Factor: Low
Complexity: Low
Popularity: Popular
Impact: Intelligence
Root Cause: Software Implementation Problems
Ease of Fix: Trivial
Description: CyberCop Scanner was able to retrieve a listing of sessions which are active on the target host. A listing of active sessions displays all resources which are currently being accessed on the target host.

Security Concerns:

Suggestion: To prevent the ability for Anonymous users to enumerate active sessions, create the following registry key:
Hive : HKEY_LOCAL_MACHINE
Key : System\CurrentControlSet\Control\LSA
Name : RestrictAnonymous
Type : REG_DWORD
Value: 1
Please note that Service Pack 3 must be installed for these restrictions to function.

26010 Enumerate RPC Bindings (EPDUMP)**Risk Factor:** Low**Complexity:** Low**Popularity:** Popular**Impact:** Intelligence**Root Cause:** Insecure Design**Ease of Fix:** Infeasible**Description:** This check will gather information about a remote machine by walking through the table of all bound RPC endpoints and listing them. This provides some information about what RPC services are running on the machine and which are accessible remotely through IP or over SMB.**Security Concerns:** The RPC bindings contain information about the network endpoint needed to connect to an RPC service. An attacker may need this information to connect to a vulnerable RPC service to perform an attack.

The bindings list also provides an attacker with some information about what services have been installed on the machine. Enumerating the list may be used as a convenient first step for identifying machines that are running vulnerable services.

Because some RPC services are assigned TCP and/or UDP port numbers dynamically, the services may be assigned ports that are not protected by your firewall.

Suggestion: There is no known method to disable this functionality at the time of this writing. The RPC locator service runs on TCP port 135.

Ensure that this port is filtered at your firewall to prevent external users from obtaining this information.





APPENDIX D

THE CYBERCOP SCANNER VULNERABILITY DATABASE

The CyberCop Scanner vulnerability database is shown in table D.1 below. Note that, due to space restriction, only the fields necessary to describe each vulnerability in the vulnerability database, are shown. See [CYBE 02] for a complete CyberCop Scanner vulnerability database.

Table D.1: The CyberCop Scanner vulnerability database

Vuln.ID	Vulnerability name	Vulnerability description
1000	Information Gathering and Recon	
1001	Finger access control check	This check attempts to contact the finger daemon on the target-host and retrieve a list of logged in users.
1002	Finger 0@host check	This check attempts to gather user information by fingering 0@target-host.
1003	Finger Redirection Check	A frequently overlooked aspect of the "finger" information system is that many implementations support forwarding of queries, allowing a finger client to request a finger server to ask another finger server for information. This can be used to hide information-gathering attacks by obscuring the source of the attack, or to obtain access to finger servers that are protected by selective network access control. This check attempts to bounce a remote finger request through the target-host finger daemon. An attempt is made to resolve a finger query that looks like this: user@some-remote-host@target-host
1004	Finger .@target-host check	Some implementations of the "finger" information server support a little-known feature triggered by requests for the user ".". In response to this query, these servers will provide a finger client with information about users who have never logged in. These users frequently have easily guessed "default" passwords. This check attempts to gather user information by fingering .@target-host.
1005	"rusers" service check	The "rusers" ONC RPC service, much like finger, provides information about users currently logged into a Unix system. This information can be used by an attacker to obtain lists of user names to attempt brute-force password guessing attacks against, and to discover the usage patterns of the system. This check attempts to retrieve information from the rusers service on the target-host. NOTE: This check will only return a listing of users in the module output on rusers version 2.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
1006	Telnet service banner present	The telnet service banner module obtains and displays the telnet banner which is obtained from the target host when connecting to the telnet service.
1007	SMTP banner-check	This check collects the message displayed upon connection to the SMTP port of the target-host.
1008	FTP banner check	The FTP banner check attempts to gather banner information from the ftp daemon.
1009	Anonymous FTP check	This check attempts to discern whether CyberCop Scanner can access an FTP server as an anonymous FTP user.
1010	"rstatd" check	"rstatd" is an ONC RPC service that provides information about the status of a system (including uptime and usage statistics) to the public. In addition to disclosing sensitive information about the configuration and capabilities of a server, "rstatd" can also provide information that is used by some programs to generate random numbers, and can thus be used as a tool to compromise other servers on a system. This module attempts to poll information from rstatd.
1011	"X.25" gateway RPC service present	The target host was found to be running the X.25 RPC gateway service. This is indicative of the target host acting as a gateway to an X.25 packet switched network.
1012	"bootparamd" RPC service present	This check identifies the presence of rpc.bootparamd. If it is present the process will then attempt to coax the NIS domain name from the server.
1013	Gopher daemon check	This check attempts to discover if a gopher daemon is running on the target host.
1014	IRC server present	This particular check discerns whether the IRC service is present on the target host.
1016	Netstat check	Some operating systems are distributed with an Internet gateway to the "netstat" command enabled in their inetd configuration. These configurations allow arbitrary entities on the Internet to obtain the output of the "netstat" command on these machines. This information can be sensitive. This check attempts to poll netstat information from a target host.
1017	Systat check	The "systat" command provides information about the current utilization of resources on a Unix system. Some operating systems are distributed with an Internet gateway to the "systat" command, allowing arbitrary entities on the Internet to gather information from the "systat" command on remote machines. The information available from systat allows an attacker to infer the configuration of the machine, and is thus sensitive. This check attempts to poll systat information from the target-host.
1018	FSP daemon check	This check discerns whether a host is running an FSP daemon.
1019	SSH information obtained	The scanner attempts to poll information from your SSH daemon about it's configuration. The information which can be gathered remotely from an SSH daemon includes: <ul style="list-style-type: none"> o SSH Version o Host key size o Public key size o Authentication methods in use o Encryption methods in use

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
1021	ESMTP check	This module checks to see if a mailer daemon supports extended SMTP commands via ehlo.
1023	Identd username gathering	This check scans a host running ident and returns the UIDs of network daemons running on the target-host.
1024	Routing table retrieved	The routing table has been retrieved from the target host's routing daemon. This service utilizes RIP (Routing Information Protocol) to maintain an updated list of routes and routing information for the host it is running on.
1026	rpc.rquotad check	The check attempts to poll rpc.rquotad on the target-host for user quota information.
1028	rpc.sprayd check	The rpc.sprayd service is offered to administrators to determine traffic statistics on a network. An administrator can send the service a stream of packets, and is presented with statistics on the number of packets which have been received.
1032	ICMP timestamp obtained	The system time was obtained from the target host utilizing a capability present within the ICMP protocol. The ICMP protocol provides an operation to query a remote host for the current system time.
1033	ICMP netmask obtained	The netmask was obtained from the target host utilizing a capability present within the ICMP protocol. The ICMP protocol provides an operation to query a remote host for the network netmask.
1034	"rpcbind" RPC service present on high numbered port	This check attempts to determine whether the target host is running a version of rpcbind which listens on a high numbered UDP port above 32770 in addition to the standard port 111. This has been known to occur on the Solaris operating system.
1035	Finger search.**@host check	This check attempts to finger search.**@target-host and monitors output to discern if usernames are returned.
1036	WWW Web Server Version	This module returns the version of WWW server running on the remote host, if it is available.
1037	"portmapper" or "rpcbind" RPC service present	The portmapper service was found running on the target host. Since RPC services do not run on well known ports this service is used to map RPC services to the dynamic port numbers that they currently reside on. RPC client programs use this service when they make a connection to a remote RPC server.
1038	S/Key Banner Check	This check will determine if the S/Key one-time password authentication system is installed on the target machine.
1039	Ascend Configurator Identification Check	Ascend Access Servers and Routers speak a protocol over the UDP "discard" port that allows the Ascend Java "Configurator" tool to locate Ascend equipment on a network automatically. An Ascend router will respond to any network user that sends a well-formed Configurator packet with a response that includes the symbolic name of the router. Attackers can use this to pick out Ascend equipment from a network (Ascend routers may be a specific target of attack, or may indicate further network connections), and to obtain the names of these routers (which may provide information on which to base password guesses).

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
1040	Network Time Protocol server present	An NTP server was found to be present on the target host. Many Network Time Protocol servers offer detailed information on their setup, including systems which they peer with, system memory configuration, and time statistics. This module obtains information from the remote NTP server using the NTP version 3 protocol and lists the information which can be obtained from the server. Information which can be obtained via NTP includes the following: - System time statistics (uptime) - System IO statistics - System memory statistics - Time daemon peer listing
1041	Trace route to host	This module traces the route to the host being scanned in the same manner as the traceroute program in UNIX or the tracert program in Windows NT. The route information is stored to the network map file as well as being returned by the module. The network mapper uses this information to build a map of the network.
2000	File Transfer Protocols	
2001	NULL Linux FTP backdoor check	This module attempts gain root level FTP access to the target-host using a backdoor in some versions of wuftp. NOTE: Other FTP servers that do not adequately enforce username/password security may report as positive to this check.
2002	FTP - root directory write-enabled	This check determines whether the anonymous FTP root directory is either world write-enabled or write-enabled by the anonymous ftp account.
2003	FTP - ports opened in sequential order	The FTP server on the target host was found to open bound ports, utilized by the PASV feature, in sequential order.
2004	Wu-FTP "site exec" check	This module checks if it can execute system commands on an FTP server via the "site exec" command.
2005	FTP directories check	The target host's FTP service was found to contain write-enabled directories.
2006	WFTP invalid password check	This check searches for older versions of WFTP (a Windows based FTP server) which would allow access to the FTP server with any username and password. Files could then be downloaded that offer further information (enticements) that could lead to further exploits of the system.
2007	FTP - bounce attack	The target host's FTP service was found to be vulnerable to the FTP bounce attack.
2010	FTP - true path check	The true home directory was obtained from the target host's FTP service.
2011	FTP - "RNFR" file deletion vulnerability	The target host's FTP service was found to contain a vulnerability in the "RNFR" command which allows overwriting and removal of files. This vulnerability allows removal of files even when the FTP servers configuration prohibits this action.
2012	FTP file write permission check	This check searches the anonymous FTP directory hierarchy for write-enabled files.
2013	FTP chmod check	This check attempts to execute the chmod command in the FTP environment.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
2014	FTP - GNU tar check	The target host's FTP server was found to contain a version of GNU tar which allows command execution.
2016	FTP - NCSA ftpd check	This check attempts to gain privileged access to older NCSA ftp servers.
2017	FTP - Windows NT Guest FTP	The target Windows NT FTP service was found to have the 'GUEST' account enabled by default. Older versions of Windows NT were distributed with this account present, and enabled by default.
2018	FTP - PASV core dump check	The target host's FTP server was found to be vulnerable to an attack utilizing the "PASV" FTP command. By initiating a connection to the FTP service, and issuing the "PASV" command prior to logging in, the FTP service crashes, leaving behind a "core" file on some operating systems.
2019	FTP - argument core dump check	The target host's FTP server was found to be vulnerable to an attack which is initiated by issuing a "LIST" command with a large number of arguments. By issuing this command, the FTP server crashes, leaving behind a "core" file on some operating systems.
2021	FTP - quote "CWD ~root" vulnerability	This module tests for the CWD ~root bug, as described in the paper "Improving the Security of Your Site by Breaking Into it" by Dan Farmer and Wietse Venema. The ftp server bug allows remote individuals to obtain root access.
2024	FTP - password file contains hashes	The target FTP server's password file was found to contain encrypted password hashes which could be cracked by an attacker.
3000	Hardware Peripherals	
3001	Unpassworded laser jet printer check	Having a laser jet printer without a password will allow remote users/intruders to modify its configuration which can result in a denial of service attack.
3002	Unpassworded Gatorboxes check	Cayman Systems manufactures a hardware device called a Gatorbox for bridging ethernet segments and appletalk networks. By default, a Gatorbox is shipped with no password. This check determines if the target-host is an unpassworded Gatorbox.
3003	Portmaster default password check	A Livingston Portmaster is a network device for central sites with remote access and point-of-presence (POP) in-a-box applications. It is often used with PPP dialup access for ISPs with modems, ISDN, CSU/DSUs, and for routing purposes. A Livingston Portmaster comes configured with a default password of !root. If no password has been set, a remote user/intruder who enters this default password can reconfigure your Portmaster. Should the Portmaster be remotely configured to fail, the result will be a denial of service. If remote users/intruders misconfigure the routing for this network device, then more subtle mischief can be accomplished that could put the data communications through this device at risk.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
3006	Ascend Port 150 Check	Ascend Port 150 Check Ascend provides networking equipment: IP routers and multi-protocol bridges which connect over ISDN (switched-56 and frame relay, also). Recent versions of Ascend's access server add an option for remote administration via TCP port 150. Attackers can use this service to guess passwords against the router, potentially allowing them to gain remote access to the router without being logged. To disable remote management, open the System Profile and set the Remote Management parameter to No. Ascend maintains a web site at http://www.ascend.com . There is technical documentation available for their products at ftp://ftp.ascend.com/pub/Doc/
3007	HP Printer Remote Print Check	HP printers that are configured for remote network printing over IP listen for requests on port 9099 and 9100. Unauthorized clients can send raw postscript files to these ports and cause their contents to be printed, regardless of the permissions set on the printer's LPD service. If the printer is being relied on for hard-copy of security auditing logs, an attacker can disable the printer by flooding it with requests, avoiding hard-copy audit trails. Also, it is possible to telnet to the printer and change the printer IP or disable logging. It is also possible to restrict the printer to accept connections from either a list of IP addresses or a subnet range.
3008	Ascend SNMP/TFTP Configuration File Retrieval	Ascend router and access server platforms are remotely manageable via the SNMP protocol. The Ascend hooks for SNMP management include the capability to download and upload the entire configuration of the router as a text file. Ascend configuration files include the plain text passwords to the router, as well as usernames, passwords, and phone numbers for outgoing connections. The attack works by using SNMP "set" commands to initiate a TFTP transfer of the config file (using the Ascend "sysConfigTftp" MIB extension). If the attacker can execute SNMP "set" commands against the router, the configuration file can be retrieved and sensitive information compromised. This module attempts to determine whether the probed host is vulnerable to the attack without actually carrying it out. This is done by setting an arbitrary SNMP variable using an SNMP "set" command. This check may be preferable to the full check when time, bandwidth, or disk space is limited; Ascend configuration files can be quite large.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
3009	Ascend SNMP/TFTP Configuration File Retrieval (full)	Ascend router and access server platforms are remotely manageable via the SNMP protocol. The Ascend hooks for SNMP management include the capability to download and upload the entire configuration of the router as a text file. Ascend configuration files include the plain text passwords to the router, as well as usernames, passwords, and phone numbers for outgoing connections. The attack works by using SNMP "set" commands to initiate a TFTP transfer of the config file (using the Ascend "sysConfigTftp" MIB extension). If the attacker can execute SNMP "set" commands against the router, the configuration file can be retrieved and sensitive information compromised. This module attempts to determine whether the probed host is vulnerable to the attack without actually carrying it out. This is done by setting an arbitrary SNMP variable using an SNMP "set" command. This check may be preferable to the full check when time, bandwidth, or disk space is limited; Ascend configuration files can be quite large.
3010	Unpassworded Ascend router check	Ascend products are shipped with no telnet password set. Having an Ascend router without a password allows remote users/intruders to read or modify its configuration, and may allow them to sniff or redirect traffic. It could also allow them to launch attacks against other machines from the compromised Ascend router.
3011	Unpassworded Netopia router check	Unpassworded Netopia router check Netopia products are shipped with no telnet password set. Having a Netopia router without a password allows remote users/intruders to read or modify its configuration.
3012	Cisco Catalyst Port 7161 Vulnerability	The supervisor module in Cisco Catalyst switches can remotely be forced to reload, stopping the switch from forwarding traffic. While the switch will recover automatically, repeat attacks can deny service indefinitely. Cisco security notice "Cisco Catalyst Supervisor Remote Reload" notes the following switches as vulnerable: The Catalyst 12xx family, running supervisor software versions up to and including 4.29. The Catalyst 29xx family (but not the Catalyst 2900XL), running supervisor software versions up to and including 2.1(5), 2.1(501), and 2.1(502). This includes the Catalyst 2901, 2902, and 2903 switches. Catalyst 2926 switches are not affected, because the Catalyst 2926 was not released until after the software fix was made. Catalyst 2900XL switches run unrelated software, and are not affected by this vulnerability. The Catalyst 5xxx series (including the Catalyst 55xx family), running supervisor software versions up to and including 2.1(5), 2.1(501), and 2.1(502). The following versions are NOT vulnerable: Catalyst 5xxx and 29xx switches running versions 2.1(6) and later. Catalyst 12xx switches running versions 4.30 and later.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
4000	Backdoors and Misconfigurations	
4001	'Rootkit' check	'Rootkit' is the name of a popular collection of trojaned OS utilities that are used by hackers to backdoor a compromised host. There is the original rootkit, as well as versions specifically for SunOS and Linux. This check attempts to identify a trojan /bin/login program by testing the default 'rootkit' username and password.
4002	'Hidesource' check	'Hidesource' is the name of a popular collection of trojaned SunOS utilities that are used by hackers to backdoor a compromised host. Like the 'rootkit' trojan horse collection, this is a collection of utilities that replace system utilities (e.g. the login program) with versions that contain a "backdoor." This check attempts to identify a trojan /bin/login program by testing the default 'Hidesource' username and password.
4004	Port daemon check	This particular check scans your machine for port daemons installed by attackers. One popular program, the socdmini written by pluvius@io.org, is a program that accepts semicolon terminated commands and executes them on the running system.
4005	ICMP backdoor check	This check looks for common implementations of ICMP backdoors by sending out a packet and waiting for a reply.
4006	'HidePak' check	'HidePak' is the name of a popular collection of trojaned Solaris utilities that are used by hackers to backdoor a compromised host. Like the 'rootkit' trojan horse collection, this is a collection of utilities that replace system utilities (e.g. the login program) with versions that contain a "backdoor." This check attempts to identify a trojan /bin/login program by testing the default 'HidePak' login and password.
4007	Back Orifice Backdoor Check	Back Orifice is a backdoor program for Windows 9x written by a group calling themselves the Cult of the Dead Cow. This backdoor allows remote access to the machine once installed, allowing the installer to run commands, get screen shots, modify the registry and perform other operations. Client programs to access Back Orifice are available for Windows and Unix. The Back Orifice server is extendable via plug-in modules. These modules include, for example, the ability to link Back Orifice to start when another program (e.g. a web browser) is started. Other, more pernicious functions include connecting to an IRC server and announcing your IP address when Back Orifice is started. This check detects if a default configuration of Back Orifice has been installed by sending a PING request to the backdoor program on the default port using the default key.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
4008	Windows NetBus 1.x Vulnerability	NetBus is a remote administration and spy tool for Windows 95/98 or Windows NT4 consisting of a server and client. The server is installed on the host which is to be remotely administered and the client is used to access the server from a remote location. By default NetBus 1.x is accessible on ports 12345 and 12346 with no password. This module accesses port 12345 to determine if NetBus has been installed and determines if a password has been set. In addition, the module verifies whether a backdoor exists in the installed version of NetBus and whether the NetBus server can be remotely crashed. NOTE: Please note that numerous programs have been written to emulate the behavior of NetBus and simply log access attempts. A positive response could reflect the fact that such a program has been found.
4009	Windows NetBus Pro 2.x Vulnerability	NetBus Pro is a remote administration and spy tool for Windows 95/98 or Windows NT4 consisting of a server and client. The server is installed on the host which is to be remotely administered and the client is used to access the server from a remote location. NetBus Pro has improved features from its predecessor, which include a remote file manager, registry manager and application redirector, plus the ability to capture screen shots, typed characters and camera images. By default, NetBus Pro is accessible on port 20034 with no password. This module accesses NetBus on this port to determine if NetBus is installed and determine if a password has been set. NOTE: Please note that numerous programs have been written to emulate the behavior of NetBus and simply log access attempts. A positive response could reflect the fact that such a program has been found.
4010	Back Orifice 2000 Server Backdoor Check	Back Orifice 2000 is a remote administration tool often used to backdoor Windows systems. The tool is divided into a server and client allowing remote access to a host including the file system and registry. Back Orifice servers are available for Windows 95, 98 and NT while client programs to access the server are available for Windows and Unix. This module detects if a Back Orifice 2000 Server has been installed.
5000	SMTP and Mail Transfer	
5001	Sendmail Wizard check	Older versions of Sendmail contained a backdoor which allowed for remote root access with a secret password. This check is designed to discern whether the version of sendmail on the target-host has this backdoor present.
5002	Sendmail DEBUG check	The check defines whether your mailer will allow DEBUG mode. Allowing DEBUG mode is a potentially dangerous security loophole that could allow a remote user to execute arbitrary commands as root via the sendmail port.
5003	Sendmail program piped aliases check	This module collects information about sendmail aliases that are piped to programs. It is common to define aliases that pipe received mail to a program for processing. The following aliases are checked: o root o news o postmaster o majordomo o decode o admin o webmaster

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
5005	Sendmail VRFY and EXPN check	Using data collected by the information gathering modules, this module attempts to get additional user information from the SMTP port of the target host with the VRFY and EXPN commands. VRFY can be used to identify valid user accounts on the system, whereas EXPN can be used to identify the delivery addresses of mail aliases and mailing lists.
5006	Sendmail mailing to programs check	This module checks to see if a mailer running on a given IP address allows mail to programs. The module opens a connection to a given IP address on port 25, sends a HELO command and then sends the following string 'mail from: root' followed by a 'rcpt to: testing'. If that command is accepted, it is assumed that the host is vulnerable. Notes: This could report false positives since some mailers won't complain about 'rcpt to: testing' but will ignore it. Smail, for example, behaves this way.
5007	Sendmail bounce 'From:' check	The 'Bounce' module checks if a mailer running on a given host allows return addresses that appear to be from applications. That is, if it is vulnerable to an SMTP bounce attack. The module opens a connection to a given IP address on port 25, sends a HELO command and then sends 'mail from: root'. It then determines if this command is accepted, and if it is, reports the host as vulnerable. No attempt to deliver mail is made. An actual attack would consist of sending mail with a 'MAIL FROM' string in the form of: " /bin/sed '1,/^\$/d' /bin/sh" This command would be followed by a 'RCPT TO' string such that it would make the mail bounce and go back to the sender, which would then pass it through the pipe and execute the body of the message. Notes: This could report false positives since Smail and the IRIX 6.x sendmail won't complain about "MAIL FROM: /bin/sed '1,/^\$/d' /bin/sh " but will ignore it.
5008	Sendmail (8.6.9) identd check	A vulnerability in version 8.6.9 of Berkeley Sendmail allows remote users to execute arbitrary commands on vulnerable systems. This module must be run as 'root', with the system's identd daemon disabled. If the remote mailer does not support the ident protocol, the module will wait for an ident connection for several seconds before reporting a site as not vulnerable.
5009	Sendmail syslog buffer overflow check	The syslog module checks if a mailer running on a target host is vulnerable to the syslog attack. Versions of sendmail were vulnerable to this attack by overflowing a buffer within the syslog() libc routine. This vulnerability would allow remote users to execute arbitrary commands as root on the remote server.
5011	Sendmail 8.6.11/8.6.12 denial of service check	This 8.6.11/8.6.12 version check module examines available sendmail banners to determine the presence of Berkeley sendmail 8.6.11 or 8.6.12. If either one is detected, it is possible that the host is vulnerable to a denial of service attack specific to these two versions.
5013	Sendmail (8.7.5) GECOS field buffer overflow check	This module checks to see if the host is running sendmail 8.7.5. Berkeley sendmail 8.7.5 has two bugs which allow for local users to gain either default user (most often daemon) or root privileges.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
5014	Sendmail (8.8.0/8.8.1) MIME buffer overflow check	This check attempts to discern if you are running sendmail version 8.8.0 or 8.8.1. Both of these versions of sendmail have a weakness which could allow intruders to access the vulnerable system as root.
5015	Sendmail Decode alias check	Some sendmail configurations include an alias called 'decode' that pipes mail through the uudecode program. By creating and sending uuencoded data to the 'decode' alias, an attacker could, for example, place an arbitrary .rhosts file onto your system.
5016	Mail forgery check	This check attempts to define if mail can be trivially forged on a target host.
5017	Sendmail daemon mode vulnerability	This check attempts to discern if you are running sendmail version 8.7 through 8.8.2. These versions of sendmail allow local users to obtain root access by causing sendmail to execute arbitrary commands as root.
5018	Sendmail (8.8.3/8.8.4) MIME buffer overflow check	This check attempts to discern if you are running sendmail version 8.8.4 or 8.8.3. Both of these versions of sendmail have a weakness which can allow intruders to access the vulnerable system as root.
5019	Majordomo Reply-To check	This check attempts to make majordomo execute commands embedded in the Reply-To field of a request. While processing a "lists" command majordomo compares the Reply-To address against the advertise and noadvertise lists. In doing so, it may be tricked into executing a command while expanding the back-tick operator. The back-tick (`) is used by Unix to enclose executable commands in a shell command line. In this case, an expression executed in a perl program. The majordomo versions noted as being vulnerable are those versions prior to 1.94.3. Because of the way this check receives notification from majordomo (it waits for a telnet connection from the mail server machine), the check may report false negatives when scanning mail servers that are behind a firewall.
5020	Qmail Denial of Service	By sending a message with a large number of recipients, it is possible to cause Qmail 1.02 and earlier to utilize all system resources. NOTE: CyberCop Scanner CANNOT determine the version of Qmail which you are running, however CyberCop Scanner CAN detect if you are running Qmail. In the case where you are running Qmail, this vulnerability will always return positive. Ensure that you are running a version of Qmail newer than version 1.02.
5021	Sendmail Relaying Allowed	This module determines whether your mail server can be used as a mail gateway or relay. When used as a mail relay, your host may be prone to "spammers" relaying mail through your host to reach their intended audience. For example, if an outside user were to send mail formatted as being to "target%somedomain.com@yourmailserver.com" that message could be re-transmitted to the target recipient, apparently originating from your mail server.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
5023	MDaemon SMTP Server HELO Overflow	Certain versions of the MDAemon SMTP server are vulnerable to an attack that allows a remote SMTP client to crash the server. Furthermore, it may be possible to execute arbitrary commands on the host running the service. Vulnerable SMTP servers overflow a buffer when an overly-long argument is given to the SMTP "HELO" command.
6000	Remote Procedure Call Services	
6003	rpc.admind security level check	Solaris' rpc.admind is a network service designed to allow remote administration capabilities to network administrators. This daemon comes by default in insecure mode, meaning it requires virtually no authentication for remote users. This allows remote users to append or change critical system information, including user accounts. This check determines if rpc.admind is in secure mode or not.
6004	rpc.pcnfsd execution vulnerability	The target host was found to be vulnerable to a vulnerability in the "pcnfsd" RPC service which can allow an attacker to execute arbitrary commands as the super-user. NOTE: To test for the vulnerability status of this service, this module disables the "pcnfsd" service on the target host. You must restart this service if this vulnerability is returned.
6005	rpc.ugidd daemon check	This check determines whether or not we can query the remote rpc.ugidd daemon and obtain usernames. The rpc.ugidd daemon is primarily present on Linux installations and allows for mapping UID and GID numbers to usernames remotely. This would enable an attacker to query the server with a range of userids and obtain remote usernames for these userid's.
6007	rpc.yppupdated check	rpc.yppupdated is a daemon which is part of the NIS suite. It is used to update changes to NIS databases remotely. Several vendor versions of rpc.yppupdated have a serious security vulnerability which allows remote users to execute commands as root. This check determines whether your host is vulnerable to this attack.
6008	rpc.statd link/unlink check	rpc.statd (or s imply statd on some machines) is used to interact with rpc.lockd to ensure file locking keeps state on NFS servers. Many versions of rpc.statd have a vulnerability whereby they can be forced to unlink, (delete) or create files as root remotely. This check discerns whether your version of rpc.statd is vulnerable to attack. There is no method to verify whether this attack worked remotely. The scanner attempts to create a file in /tmp called CyberCop.rpc.statd.vulnerability. If this file exists on the specified host, then your host is vulnerable.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
6009	NIS domain name check	NIS (Network Information System) does most of its authentication by having the client pass the server the NIS domain name as a password. When a client provides the correct NIS domain name, it may request NIS maps. Often an NIS domain name is easily guessable. If this is the case then a user anywhere on the Internet who knows your NIS domain name may request your maps - Passwd.byname, for example. Note that newer versions of NIS require the client to belong to an ACL (Access List), such as securenets.
6014	rpc.selection_svc check	The target host was to be running a vulnerable version of the selection_svc RPC service. This service contains a security vulnerability which can allow an attacker to read any arbitrary file on the target system.
6015	rpc.rwalld check	The rwall daemon is a service which will broadcast messages from remote hosts to all users who are logged into the system. While it is useful for sending broadcast messages across an entire network for administrative purposes, it lacks proper authentication. This provides an attacker with the ability to send messages to every user logged into your servers. This also allows an attacker to flood users with messages.
6016	Portmapper spoofed register/unregister	The portmapper, which provides service to translate port numbers for RPC services, has a number of weaknesses. One of these weaknesses allows remote users to register/unregister services on a remote host by way of forging UDP packets. An attacker can utilize this to gain increased access to the local machine. An example attack involves unregistering a service from the portmapper, and then re-registering the service on a new port, which they have control over. This allows an attacker to impersonate security critical services and gain increased access to the network. Some versions of ONC RPC for Microsoft Windows NT are also known to contain this vulnerability.
6019	Mount & NIS services on non-reserved ports check	This module checks for mount daemon and NIS services running on non privileged ports. Any of the above services running on non-reserved are most likely vulnerable to port hijacking. If a user can hijack these services, he can then intercept or supply data from or to client programs.
6020	Portmapper register/unregister check	This module determines whether attackers can register and unregister services on your portmapper/rpcbind by using standard RPC calls. This vulnerability does not require address forgery to succeed and provides any network user with the ability to register new services and unregister existing services. Some versions of ONC RPC for Microsoft Windows NT are also known to contain this vulnerability. BSDI 2.1, 3.0 and Ultrix are known to be vulnerable to this attack.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
6021	Portmapper register/unregister through callit	This check determines if portmapper services can be set and unset by utilizing a feature within the portmapper/rpcbind program known as callit(). The callit() function allows forwarding of requests to local services as though they were coming from the local system itself. This allows attackers to bypass IP address based authentication checks, to register and un-register services, in addition to exploiting other services. This check attempts to register a new service on the portmapper/rpcbind by utilizing this technique. In this way the set request appears to come from the local machine and may bypass address checks.
6025	Sequential port allocation check	This check is designed to test if a host will spawn its listening ports in sequential order. If this is the case, attackers can implement host spoofing techniques to services which poll other hosts for authentication. Examples of such services include, for instance, any service which requires authentication from DNS servers.
6027	rpc.ttdbserver buffer overflow vulnerability	The ToolTalk service allows independently developed applications to communicate with each other by exchanging ToolTalk messages. Using ToolTalk, applications can create open protocols which allow different programs to be interchanged, and new programs to be plugged into the system with minimal reconfiguration. The ToolTalk database server (rpc.ttdbserverd) is an ONC RPC service which manages objects needed for the operation of the ToolTalk service. ToolTalk-enabled processes communicate with each other using RPC calls to this program, which runs on each ToolTalk-enabled host. This program is a standard component of the ToolTalk system, which ships as a standard component of many commercial Unix operating systems. The ToolTalk database server runs as root. Due to an implementation fault in rpc.ttdbserverd, it is possible for a malicious remote client to formulate an RPC message that will cause the server to overflow an automatic variable on the stack. By overwriting activation records stored on the stack, it is possible to force a transfer of control into arbitrary instructions provided by the attacker in the RPC message, and thus gain total control of the server process.
6028	rpc.rexd check	This check attempts to exploit a weakness in rpc.rexd. The weakness in question is that common implementations of rexd take their authentication from the client. This allows remote users to execute commands remotely with any other UID (User ID) than root.
6034	nfsd port 4045 Check	This check attempts to determine whether the target host is running a version of lockd which listens on port 4045 and is capable of servicing NFS requests.
6035	SGI fam server check	This check attempts to obtain a list of files from the SGI fam service.
6036	rpc.statd Bounce vulnerability	A vulnerability in the rpc.statd service provides attackers with the ability to "bounce" RPC calls through this service. Using this technique, an attacker has the ability to pass a packet, as though it were coming from the local system, including over the loopback interface.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
6037	Solaris automountd vulnerability	This module checks for a vulnerability in the automount daemon on Solaris systems. This vulnerability can allow local users to obtain increased access to the target host. This vulnerability can also be combined with a vulnerability present in the rpc.statd service, to exploit automountd remotely.
7000	Networked File Systems	
7001	NFS - Superfluous server check	The target host was found to have an NFS server running without any directories being exported. Many systems enable NFS by default, and it is not uncommon to see such machines running NFS when they are not, in fact, importing or exporting anything. The NFS service is quite complex and has a long history of security problems. If it is not necessary for your system to run NFS, you should consider disabling the service.
7002	NFS - world exports found	The target host was found to have directories exported to "everyone" via NFS. By exporting directories to "everyone", anyone who can connect to the target host is able to access these file systems.
7003	NFS - exporting out of administrative scope check	The target host was found to be exporting file systems via NFS to hosts which are outside of the target host's network. You should ensure that your security policy permits exporting of file systems outside of the host's local network.
7004	MOUNTD - proxy mount vulnerability	Older portmappers were flawed in as much as they would forward requests from other services on remote hosts, through itself via the callit procedure. When the portmapper forwarded these requests the source address for the request becomes that of the localhost. This attack can be used to talk mountd into mounting file systems to hosts which it does not trust in its /etc/exports file. This check determines whether your portmapper has this problem.
7005	MOUNTD - exported file system list retrieved	A list of exported file systems was retrieved from the target host. An attacker may utilize this list to infer a trust relationship on the network, as well as discover file systems which may be exported without restrictions.
7006	NFS - exporting sensitive file check	Exporting sensitive directories can open yourself up to a number of attacks provided an attacker can mount your file system in order to either read or write to these directories.
7007	NFS - fake UID check	Older mount daemons could be fooled into providing access under any UID provided an attacker could perform a mount. This check defines if your daemon has this problem.
7008	NFS - mknod check	Some older NFS servers will allow for users to mknod (create) device files on NFS mounted file systems. This could allow a cracker to create a kmem device which was writable that he/she could then use to swap their UID to 0 (root). This check attempts to exploit this problem.
7010	NFS - unchecked cd .. check	Some older mount daemons did not effectively restrict access to mounted file systems. This particular flaw allowed users to cd .. back up the directory tree onto the non exported file system.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
7011	MOUNTD - Ultrix/OSF remount check	Some versions of Ultrix and OSF mount daemons allowed for users outside of their exports list to mount file systems. This check discerns if this problem is present on a target host.
7013	MOUNTD - exports list over 256 characters check	On some mount daemons if the export list is over 256 character long it will allow anyone to mount your NFS shared directories regardless of whether they are in the exports list or not. This check sees if your export list is over 256 character long, and attempts to mount those file systems.
7014	MOUNTD - Linux/Solaris file existence vulnerability	Linux and Solaris operating systems allow remote user to determine the existence of files on the remote server via rpc.mountd, the NFS mount daemon. By analyzing the error messages returned by the rpc.mountd daemon, an attacker can determine whether files exist, without legitimate access to the NFS server. NOTE: This module may report a false positive on systems that export /etc via NFS.
8000	Denial of Service Attacks	
8001	Echo/chargen packet flood check	The character generator (chargen) service is designed to simply generate a stream of characters. It is primarily used for testing purposes. Remote users/intruders can abuse this service by exhausting system resources. Spoofed network sessions that appear to come from that local system's echo service can be pointed at the chargen service to form a "loop." This session will cause huge amounts of data to be passed in an endless loop that causes heavy load to the system. When this spoofed session is pointed at a remote system's echo service, this denial of service attack will cause heavy network traffic/overhead that considerably slows your network down. It should be noted that an attacker does not need to be on your subnet to perform this attack as he/she can forge the source addresses to these services with relative ease. Denial of Service (DoS) attacks are usually easy to accomplish and harder to mitigate. Often the vulnerability is presented in the operating system (OS) feature implementation (i.e. IP packet handling) or application software bug (i.e. improper boundary checking, resource limitations, or untested interactions) The main defenses against DoS attacks are: - maintain -- apply appropriate vendor functionality and security patches to reduce the risk - minimalism -- remove unnecessary services and functionalities to remove a Dos attack through that vector - harden -- to have configured your system with enough resources - to withstand that attack - to "raise the bar" on the attacker and make it require more effort to be successful - monitor -- to have and monitor audit trails, logs and monitoring programs to discover the attack

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
8002	Recursive finger check	Older finger daemons supported "gatewaying" the finger command whereby a user could finger a user@someotherhost@thathost, though this was not common. Of greater concern, though, is the fact that the finger daemon could be forced to do recursive searches if a remote user submitted a large number of '@' symbols before the hostname (e.g finger @@@@thathost). If you are running a vulnerable finger daemon, this recursive searching can force your machine to fill the process table with recursive searches. In theory, if enough @'s are supplied it will force the machine to swap out physical memory to virtual memory, eventually causing the system to utilize all available memory for this task. Denial of Service (DoS) attacks are usually easy to accomplish and harder to mitigate. Often the vulnerability is presented in the operating system (OS) feature implementation (e.g. IP packet handling) or application software bug (i.e. improper boundary checking, resource limitations, or untested interactions) The main defenses against DoS attacks are: - maintain -- apply appropriate vendor functionality and security patches to reduce the risk - minimalism -- remove unnecessary services and functionalities to remove a Dos attack through that vector - harden -- to have configured your system with enough resources - to withstand that attack - to "raise the bar" on the attacker and make it require more effort to be successful - monitor -- to have and monitor audit trails, logs and monitoring programs to discover the attack
8003	Solaris rpcbind kill check	Due to a bug in Solaris's libnsl up to 2.5 an attacker can force rpcbind to stop offering single service lookups. In effect, any remote client querying a remote server which is run out of rpcbind, will not be able to connect to the application being served.
8004	SYN flood check	A common and dangerous denial of service of attack is called SYN flooding. This attack can be used to completely disable your network services by flooding them with connection requests. This will fill the queue which maintains a list of unestablished incoming connections, forcing it to be unable to accept additional connections.
8005	ICMP unreachable check	A common denial of service attack is to send ICMP unreachable packets from a spoofed address to a host. This causes the host being hit with the packets to tear down all legitimate TCP connections with the host which is being spoofed in the ICMP packet.
8006	Routed append check	Most route daemons which are based off of generic Berkeley source code have a bug which will allow remote users to append garbage over system critical files. If this module returns vulnerable, it does not necessarily mean that your host is vulnerable to this attack. The scanner has attempted to create a file in /tmp called Cybercop.in.routed.vulnerability. There is no method for the scanner to determine whether this file was successfully created. Please check the /tmp directory on this host for the existence of this file.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
8007	Linux inetd check	On some Linux hosts if a SYN packet is sent and immediately followed by an RST packet, it will kill inetd(8) on the target host.
8008	SunOS 4.1.3 UDP reboot check	Unpatched versions of SunOS 4.1.3 can be forced to reboot if given a UDP packet with bizarre options set.
8009	In.comsat check	The comsat daemon is a program which watches for incoming mail, and notifies a user of newly arrived mail. The problem with comsat is that it can be fooled into issuing endless messages, resulting in a denial of service attack to users.
8010	PASV denial of service check	The PASV command in FTP servers asks the server machines to open a port and return this port number to the client. The problem is that many FTP servers will allow a user to continuously issue PASV commands spawning open ports until there are none left.
8011	Portmaster reboot check	Older portmasters could be forced to reboot if sent packets with particular commands in them.
8012	Compaq/Microcom 6000 Denial of Service check	Compaq/Microcom 6000 Denial of Service check Certain versions of Compaq's Microcom 6000 Remote Access Concentrator CPS is susceptible to a denial of service attack, which will make it unable to accept telnet connections, until it's restarted.
8016	Syslog write check	This check has CyberCop Scanner attempt to write information to your syslog daemon. If successful it indicates an attacker could write enough erroneous data to your syslog file to fill your log files and cause hard disk failure.
8017	PING denial of service attack	Many unix variants are prone to an attack whereby a remote user can cause your system to reboot or panic by sending it an oversized packet. This is performed by sending a fragmented packet larger than 65536 bytes in length, causing the remote system to incorrectly process this packet. The result is that the remote system will reboot or panic during processing. This problem is widely known as the "Ping of Death attack".
8019	Serv-U FTP server CWD overflow	This check determines whether you can crash the Win95 Serv-U ftp server by sending it a request to change directories to a directory whose name is longer than 256 characters. It is likely, but not verified, that this can also be used to remotely execute arbitrary commands on the ftp server.
8020	Ascend/3com router zero-length TCP option DOS	This check determines whether you can reboot an ascend router by sending it a TCP packet with a zero-length TCP option. There are several widely distributed programs which make it easy for people to carry out this attack.
8023	Windows NT - Out Of Band data DOS	This check determines whether your Windows 95 or Windows NT servers are vulnerable to a denial of service attack utilizing out of band data. By connecting to the NetBIOS port (139) on Windows 95 and Microsoft Windows NT systems, it is possible to crash the system by sending out of band data on the connection.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
8024	IRC Daemon Denial of Service	IRC (Internet Relay Chat) allows realtime conversation and discussion on the internet. A vulnerability exists in some IRC server versions which allow a malicious user to crash the server. This leads to a denial of service attack which prevents users from connecting to the server.
8025	Ascend port 150 crash	Ascend routers are prone to a denial of service attack, whereby a malicious user can crash the router or terminal server by connecting to the remote administration port (150) and entering the correct data.
8026	CISCO Web Server DOS	Many current versions of CISCO IOS have the ability to allow configuration via a built in WWW server on the router or terminal server. This web server contains a serious vulnerability which allows an attacker to crash the device by specifying an abnormally long URL.
8027	Solaris syslogd Crash	Certain versions of Solaris syslogd will crash when they receive a syslog message off the network from a host without inverse DNS entries. This allows an attacker to disable security auditing before attacking a host, avoiding detection by programs like TCP wrappers. This module attempts to determine if the host is vulnerable to this problem by forging a syslog request from a host without inverse entries. If the host is vulnerable, it's syslogd will be disabled, and must be re-started via administrative intervention.
8028	Rwho Daemon Buffer Overflow	This module determines whether the rwho daemon running on the target host is vulnerable to a buffer overflow, allowing remote users to kill off the daemon. The rwho daemon gathers information on other systems running on the same subnet. By sending a fake rwho request with an overly long hostname present, it is possible to cause the daemon to fault, disabling gathering of accurate network information. This problem is not known to lead to further system access. The buffer overflow is only known to disable this service.
8029	IIS Long URL Denial of Service	Microsoft IIS WWW server version 2.0 and version 3.0 are vulnerable to a denial of service attack, allowing a user who specifies a long URL, to crash the server. By mishandling this long URL, the WWW server faults, crashing the server, therefore disabling all WWW services on the host.
8030	Windows NT - Messenger Service Denial of Service	The messenger service is a service which is used by Windows NT systems to send notification messages to users on the system. This service is commonly used to send messages regarding events such as security alerts, and print job status. By sending a message with an abnormally long username to the messenger service, it is possible for an attacker to disable this service, and prevent the user who is logged into the system from receiving any further notifications.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
8031	Windows NT - SMB Denial of Service	Microsoft Windows NT systems prior to Service Pack 3 contain a serious security vulnerability which can allow a remote user to cause the server to crash with a blue screen. By connecting to the SMB port (TCP port 139) and attempting to execute a SMB file command, prior to logging in and/or accessing any shares, the system will crash.
8032	LAND Denial of Service attack	A denial of service present in many operating systems, this attack allows a malicious user to completely disable the target host by sending a single TCP packet. This attack is performed by sending a TCP packet to a running service on the target host, with a source address of the same host. The TCP packet is a SYN packet, used to establish a new connection, and is sent from the same TCP source port, as the destination port. When accepted by the target host, this packet causes a loop within the operating system, essentially locking up the system.
8033	Windows NT - Fragment Denial of Service attack	The NT TCP/IP stack uses a faulty reconstruction algorithm to reconstruct fragmented IP packets. This has a number of effects including allowing packets to be reconstructed without ever receiving the first fragment and allowing an attacker to corrupt the memory of the TCP/IP stack. Because firewalls often only filter the first fragment of an IP packet, the first effect can allow an attacker to send packets through a firewall unfiltered. The second effect allows an attacker to crash an NT system by sending carefully crafted packets that corrupt the TCP/IP stacks memory.
8034	Windows NT - LSASS.EXE Denial of Service	A vulnerability within the LSASS.EXE process on Windows NT systems allows for a denial of service attack, which causes an Access Violation in LSASS.EXE. This denial of service causes the LSASS.EXE process to stop running, preventing logons from the console, as well as preventing Event Viewer and Server Manager from operating.
8035	Windows NT - RPCSS.EXE Denial of Service	A vulnerability in the RPCSS.EXE process on Windows NT systems allows for a denial of service attack. This denial of service attack causes the RPCSS.EXE process to run in an infinite loop driving the system CPU usage up to 100%. In addition the RPCSS process stops responding to requests.
8036	Windows NT - IIS ..\.. Denial of Service	The Windows NT IIS Server running on the target host is vulnerable to a denial of service attack, allowing malicious users to crash the IIS server. If the CyberCop Scanner Security Auditing System has discovered this vulnerability present on the target host, this attack has been successfully launched, and the system should be restarted.
8038	IP Fragmentation/Teardrop Attack	This module sends out invalid fragmented IP packets that trigger a bug in the IP fragment reassembly code of some operating systems. This vulnerability allows an attacker to crash the target system, resulting in loss of service. Due to the nature of this attack, this module is not reliable. In some instances the target host will not crash immediately after this attack has been launched. The second variation of this attack (Teardrop 2) has been verified to work 100% against vulnerable systems. The second variation is located in module 8039.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
8039	IP Fragmentation/Teardrop-2 Attack	This module sends out invalid fragmented IP packets that trigger a bug in the IP fragment reassembly code of some operating systems.
8040	Cisco 760/766 Access Router "login" DOS	Cisco 760-series routers are remote access routers for ISDN connections. Due to an implementation problem, they are vulnerable to an attack that can cause the router to crash and reboot. The attack works by responding to the router's "Password" prompt with an overly-long random string. This overflows a buffer in the router, which subsequently crashes. This module attempts to determine whether a remote system is vulnerable to attack by connecting to the router's "telnet" port and sending an overly-long password. If the test is successful, the router will crash and reboot; if not, the router will remain stable throughout the test. Due to the nature of this problem, it is possible that it (like many buffer overflow bugs similar to it) can be exploited to obtain access to the router remotely. This has not yet been confirmed publicly.
8041	IP-Switch IMail / Seattle Labs Sendmail VRFY Overflow	Certain versions of the SMTP mail servers from the IP-Switch IMail package and the Seattle Labs Sendmail package are vulnerable to an attack that causes the mail server software to crash. This allows an attacker to compromise the availability of the mail service on vulnerable systems. The attack works by sending an overly-long email address in conjunction with an SMTP "VRFY" (verify email address) command. In vulnerable software, this causes a buffer overflow to occur, which in turn causes the mail software to crash. This module attempts to ascertain the vulnerability of a remote mail server by sending an overly-long SMTP "VRFY" command to the mail server. If the probe is successful, the mail service will crash. If not, the service will remain stable throughout the probe. Due to the nature of this vulnerability, it is possible that it (like other buffer overflow bugs) can be exploited to obtain remote access to the mail server. This has not been confirmed publicly.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
8042	Ascend "discard" Service DOS	Ascend routing and access server platforms, including the Pipeline, MAX, and TNT systems, are vulnerable to a denial of service attack that allows arbitrary remote users to reboot the machine. While the machine is in the process of rebooting, it will be unable to forward traffic, and any connections (modem, ISDN, etc) will be dropped. Sites that rely on Ascend routing hardware for connectivity can be cut off from the network with this attack. The attack works by sending a specially formatted packet to the UDP "discard" service on the router. Ascend hardware speaks a special proprietary "configurator" protocol over UDP "discard", and when the system receives a malformed configurator packet, it crashes and reboots. Any attacker that can send packets to the "discard" port of a vulnerable Ascend router can thus crash and reboot it. This module attempts to crash an Ascend router by sending a malformed configurator packet to the router. If the attack is successful, the router will crash and reboot. If not, the router will remain stable during the probe.
8043	rpc.statd buffer overflow	This module checks for a vulnerability in the rpc.statd service present on NFS client and server systems. A buffer overflow vulnerability present in this service allows execution of arbitrary commands on vulnerable systems.
8044	Microsoft RAS PPTP DOS	Microsoft provides remote access capabilities to Windows NT machines via its RAS subsystem. In order to provide remote network access with enhanced security, RAS uses a Microsoft proprietary protocol called PPTP (Point-to-Point Tunneling Protocol). In a typical configuration, arbitrary clients on the Internet have the ability to speak a limited amount of PPTP to a RAS server. Due to an implementation problem in Microsoft's code, it is possible for an attacker to cause a RAS server to crash by sending a specific type of PPTP request to the server with a malformed packet header field. This can be used by an attacker to deny legitimate remote access to the RAS server.
8046	Cisco IOS remote router crash check	A software implementation bug in Cisco IOS makes it possible for an attacker that gains access to a "login" prompt to cause the IOS device to crash and reload. Access to the login prompt can be obtained over TCP/IP, asynchronous lines, a local console connection or any other connection supported by Cisco IOS. This check, however, only tries to establish a connection to the login prompt over a TCP stream. If the test is successful, the router will crash and reboot; if not, the router will remain stable throughout the test.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
8049	WinGate Proxy Connection Loop DOS	WinGate is a popular Windows software proxy package. In some configurations, WinGate servers allow arbitrary users to connect to the command-line interface of the server; this enables arbitrary people to bounce connections off the proxy. This is a vulnerability, as it can be used to launder connections used for attacks. See check 13013 for more information. Some versions of the WinGate proxy server are susceptible to a denial-of-service attack which allows a remote attacker, who has authorization to connect to the WinGate command line interface, to render the server nonfunctional. This attack is carried out by continuously requesting the server to connect to itself, via the localhost interface, until the server runs out of memory to handle further requests.
8050	Xylogics/Bay Annex Ping CGI Overflow	Bay Networks, a Nortel Networks subsidiary, acquired and supports a terminal server solution from Xylogics called an Annex server. Annex servers allow remote users to obtain dialup connections to a network; they also potentially allow network clients to dial out of the network, and are thus coveted targets for attackers. Some versions of the Annex software are susceptible to a denial of service attack involving the server's built-in web server. Vulnerable Annex versions support a "ping" CGI program which, when fed overly- long queries, overflows an internal buffer and disables the entire access server. The full extent of this vulnerability is not known. Typically, overflow conditions that result in denial of service can be exploited to obtain complete access to the afflicted software, which can then be used as a launching point for further attacks.
8051	HP LaserJet 5 SNMP Denial of Service	
8053	Windows NT - SLmail v3.1 Denial of Service check	Builds of Seattle Lab's SLMail 3.1 smtp service (slsmtp.exe) prior to build number 2961 are susceptible to a denial of service attack. This attack will raise the CPU usage of the slsmtp.exe process to almost 100%.
8054	BSD Option Fragmentation Vulnerability	The IP fragment reassembly algorithm in BSD derived implementations incorrectly reassembles fragments containing invalid IP options with the potential to crash or hang vulnerable systems. Vulnerable systems include BSDI, FreeBSD, and OpenBSD.
9000	Password Guessing/Grinding	
9001	FTP Password Guessing	This module attempts to guess passwords via the FTP server. A common security problem are networked hosts with easily guessable usernames and passwords. In some instances, operating systems come pre-configured with several default user accounts which can allow access to anyone. CyberCop Scanner will attempt to login to the remote server with a list of usernames and passwords which are stored in the files "userlist.txt" and "passlist.txt" by default. CyberCop Scanner will also save any usernames which can be obtained via finger, rusers and other services and attempt to login as those users.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
9002	Telnet Password Guessing	This module attempts to guess passwords via the telnet daemon. A common security problem is having networked hosts with easily guessable usernames and passwords. In some instances, operating systems come pre-configured with several default user accounts which can allow access to anyone. CyberCop Scanner will attempt to login to the remote server with a list of usernames and passwords which are stored in the files "userlist.txt" and "passlist.txt" by default. The scanner will also save any usernames which can be obtained via finger, rusers and other services and attempt to login as those users.
9003	POP Password Guessing	This module attempts to guess passwords via the POP server. A common security problem are networked hosts with easily guessable usernames and passwords. In some instances, operating systems come pre-configured with several default user accounts which can allow access to anyone. CyberCop Scanner will attempt to login to the remote server with a list of usernames and passwords which are stored in the files "userlist.txt" and "passlist.txt" by default. CyberCop Scanner will also save any usernames which can be obtained via finger, rusers and other services and attempt to login as those users.
9004	IMAP Password Guessing	This module attempts to guess passwords via the IMAP server. A common security problem are networked hosts with easily guessable usernames and passwords. In some instances, operating systems come pre-configured with several default user accounts which can allow access to anyone. CyberCop Scanner will attempt to login to the remote server with a list of usernames and passwords which are stored in the files "userlist.txt" and "passlist.txt" by default. CyberCop Scanner will also save any usernames which can be obtained via finger, rusers and other services and attempt to login as those users.
9005	Rexec Password Guessing	This module attempts to guess passwords via the rexec daemon. A common security problem is having networked hosts with easily guessable usernames and passwords. In some instances, operating systems come pre-configured with several default user accounts which can allow access to anyone. CyberCop Scanner will attempt to login to the remote server with a list of usernames and passwords which are stored in the files "userlist.txt" and "passlist.txt" by default. CyberCop Scanner will also save any usernames which can be obtained via finger, rusers and other services and attempt to login as those users.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
9006	Rlogin Password Guessing	This module attempts to guess passwords via the rlogin daemon. A common security problem are networked hosts with easily guessable usernames and passwords. In some instances, operating systems come pre-configured with several default user accounts which can allow access to anyone. CyberCop Scanner will attempt to login to the remote server with a list of usernames and passwords which are stored in the files "userlist.txt" and "passlist.txt" by default. CyberCop Scanner will also save any usernames which can be obtained via finger, rusers and other services and attempt to login as those users.
9007	Password(s) guessed via WWW server	CyberCop Scanner was able to guess the username and password of a valid account which is utilized to obtain privileged access to the target WWW server.
10000	World Wide Web, HTTP, and CGI	
10001	NCSA WebServer buffer overflow check (versions 1.4.1 and below)	NCSA's web server software prior to version 1.4.1 had a buffer overflow that could be exploited to give a remote user access to the server. This check will attempt to exploit the buffer overflow in NCSA httpd.
10002	test-cgi check	Some HTTP servers ship with a CGI (Common Gateway Interface) script called test-cgi. This script can be subverted to list files and directories, anywhere on the host machine. This check searches for the test-cgi script and determines whether directories can be listed remotely.
10003	WWW Perl check	The WWW Perl check searches your cgi-bin directory for executable implementations of Perl. Many web server administrators inadvertently place copies of the Perl interpreter into their web server script directories.
10004	WWW phf check	The phf CGI program is a gateway to the "PH" phone book system, which is frequently used at Universities to provide online student phone books. The phf web gateway improperly parses incoming web requests when they contain quoted newline characters, allowing attackers to submit requests that will cause phf to execute an arbitrary command on the web server. This check searches for the phf script and attempts to exploit it.
10006	Microsoft .bat/com check	Some WWW servers, notably WebSite (an O'Reilly & Associates web server for Windows NT) and Microsoft's IIS (Internet Information Server) Web Server have a weakness which allows users to execute arbitrary commands with '.bat' or '.cmd' files. This check searches for such files and attempts to exploit them.
10008	Shell interpreter check	Leaving executable shells in your cgi-bin directory can enable remote users to execute arbitrary commands on your host, as the UID which owns the shells. This can lead to your machine being breached. This check looks for the following shells in your cgi-bin directory: * ash * bash * csh * ksh * sh * tcsh * zsh
10009	PHF bash vulnerability	A vulnerability in the GNU BASH shell allows usage of characters with a decimal value of 255 as command separators. This problem allows users to send command strings to remote servers and have the remote server execute them.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
10010	WWW finger check	Some web sites implement a web gateway to the "finger" service, allowing remote web clients to execute finger queries against arbitrary hosts. In environments where the "finger" service has been determined to be a security risk (due to the sensitivity of the information it provides), a web finger gateway can be used to execute finger queries against the server, allowing an attacker to obtain information about its users. This check attempts to find a web-based finger gateway and execute it.
10012	WWW Server is not running in a "chroot" environment	The target WWW server was found to not be running in a "chroot" environment. When running in a "chroot" environment, the WWW server's file system is limited to a small subset of the host's real filesystem. The target WWW server has the ability to access the entire file system on the target host.
10014	NCSA WebServer buffer overflow check (version 1.5c)	NCSA's web server software prior to version 1.5c had a buffer overflow that could be exploited to give a remote user access to the server. This check will attempt to exploit the buffer overflow in NCSA httpd.
10015	Nph-test-cgi check	Many Unix-based web servers are bundled with a sample CGI program called "nph-test-cgi". nph-test-cgi is a test script that allows "non-parsed headers" to be sent via HTTP. Due to improper quoting of request parameters, attackers can formulate requests to this program that will cause it to list all files on the system.
10016	AnyForm CGI check	AnyForm is a CGI program that allows webmasters to create arbitrary form submission pages without writing a dedicated CGI program for each form. AnyForm runs the Bourne shell to execute Sendmail, which it uses to send form results to the web administrator. Due to improper quoting of form field parameters, an attacker can place shell metacharacters in form fields, which will cause AnyForm to execute an arbitrary command on the web server. This check searches for the AnyForm script and attempts to exploit it.
10017	FormMail check	FormMail is a CGI program that allows the creation of arbitrary form submission web pages without writing a dedicated CGI program for each. FormMail executes the Bourne shell in order to run a mail program, which is used to send form results to the web administrator. Due to improper quoting of form fields, an attacker can place shell metacharacters in a form field, forcing FormMail to execute an arbitrary command.
10018	ScriptAlias check	The ScriptAlias check attempts to exploit a problem inherent in both NCSA httpd (all versions up to and including 1.5) and Apache httpd prior to 1.0. The problem is that configuring a ScriptAlias directory within the Document Root permits users to retrieve a CGI program rather than execute it. This will allow remote users to download scripts instead of executing them. In effect this will give the attacker the ability to search your CGI forms for weaknesses and or steal proprietary programs.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
10019	Guestbook CGI	The Guestbook CGI program allows web browsers to leave their name in an electronic guestbook. If the web server implements the Server-Side Includes (SSI) extension, the Guestbook program can be used to execute an arbitrary command on the web server, by leaving a name and message that includes HTML tags for an SSI command.
10020	Test-cgi "*" check	Some HTTP servers ship with a CGI script titled test-cgi. This script can be subverted to list files and directories, anywhere on the host machine. Later versions of the test-cgi script, which were meant to prevent the use of wildcards to obtain file listings have a bug which allows people to obtain file listings using "*" instead of "**".
10021	Nph-test-cgi "*" check	Some HTTP servers ship with a CGI (Common Gateway Interface) script titled nph-cgi-test. This script can be subverted to list files and directories, anywhere on the host machine. This check searches for the nph-test-cgi script and attempts to exploit it using "*" instead of "**".
10022	Apache httpd cookie buffer overflow	Version 1.1.1 and earlier of the Apache httpd have a remotely exploitable buffer overflow in their cookie generation code. This check determines whether you are running version 1.1.1 of the Apache httpd with the cookies module enabled. If you are vulnerable to this attack, remote individuals can obtain access to your web server machine.
10023	Windows NT - WebSite buffer overflow	Version 1.1e of the WebSite web server for Windows NT contains a serious vulnerability allowing remote users to execute arbitrary commands on systems running WebSite for Windows NT. The vulnerability exists in the example CGI program which is located in /cgi-shl/win-c-sample.exe which contains a buffer overflow. This allows an attacker to specify instructions for the web server to execute, enabling them to execute any Windows NT command.
10024	Windows 95 - WebSite buffer overflow	The release version of the WebSite web server for Windows 95 contains a serious vulnerability allowing remote users to execute arbitrary commands on systems running WebSite for Windows 95. The vulnerability exists in the example CGI program which is located in /cgi-shl/win-c-sample.exe which contains a buffer overflow. This allows an attacker to specify instructions for the web server to execute, enabling them to execute any Windows 95 command.
10025	php.cgi file printing bug	PHP is a CGI program that allows highly flexible dynamic web pages to be created, by feeding web pages through an interpreter. The PHP interpreter reads input files, executes PHP commands, and sends the output to web clients. As distributed, it is possible for an attacker to request an arbitrary file from PHP, rather than a specifically allowed web page. Misconfigured PHP programs will allow an attacker to read any file the web server can read.
10026	php.cgi buffer overflow	php.cgi 2.0beta10 and earlier suffer from a command line buffer overflow which makes it possible for a remote attacker to obtain access to your web server.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
10027	SGL wrap CGI	The version of /var/www/cgi-bin/wrap shipped with some versions of IRIX permits users to obtain listings of any directory on your system which ordinary users can read. In addition, the default inetd.conf instructs IRIX to place a web server on port 8778 as well as port 80.
10028	IRIX /cgi-bin/handler check	The /cgi-bin/handler program, shipped with Irix 6.2, makes it possible for remote individuals to execute arbitrary shell commands.
10029	Glimpse HTTP check	Glimpse is a search engine used to efficiently search for information in large numbers of files. "aglimpse" is a CGI program that makes up part of a WWW gateway to Glimpse. A vulnerability exists in the /cgi-bin/aglimpse script which allows a remote user to execute arbitrary commands on the remote system as the user which the web server runs as.
10030	GAIS websendmail check	WEBGAIS is a search tool. Some older versions of the WEBGAIS tool is bundled with a CGI program called "websendmail", which allows form input to be mailed to an administrator. The "websendmail" CGI program improperly processes information from form fields, and allows them to contain shell metacharacters. This can be used to coerce the program into executing an arbitrary program on behalf of an attacker.
10031	WebSite Uploader CGI check	Uploader.exe is a sample CGI script that comes with O'Reilly's WebSite web server for NT. Due to insufficient argument checking, the uploader CGI program will allow attackers to upload files to arbitrary directories under the web server root directory. This module uploads a text file to one of the CGI directories. An attacker could upload a CGI script and invoke it to get access to the web server.
10032	PHP mlog Example Script Check	PHP is a CGI program that allows administrators to easily and flexibly create dynamic web pages. PHP-enabled web pages are fed through the PHP interpreter, which executes commands embedded in the web pages and feeds the output to web clients. The PHP scripting language contains an example script called mlog.phtml which, due to insufficient checking of a script argument, will allow a user connecting via WWW to read any file readable by the web server daemon. This check tries to obtain the password file in /etc/passwd using this script.
10033	PHP mylog example script test	PHP is a CGI program that allows administrators to easily and flexibly create dynamic web pages. PHP-enabled web pages are fed through the PHP interpreter, which executes commands embedded in the web pages and feeds the output to web clients. The PHP scripting language contains an example script called mylog.phtml which, due to insufficient checking of a script argument, will allow a user connecting via WWW to read any file readable by the web server daemon. This check tries to obtain the password file in /etc/passwd using this script.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
10034	Cisco HTTP Server Presence	Newer Cisco routers can be configured through a web interface that works via an HTTP server in the router software. It is possible that the presence of this server can allow an attacker to gain extended access to a router. Presence of this server also indicates an out-of-the-box configuration of the router which may be vulnerable to other attacks.
10035	wwwcount Stack Overflow Check	Certain versions of Muhammad Muquit's wwwcount counter CGI program are vulnerable to a stack overrun caused by the processing of an overly-large query string. Attackers can exploit this problem to run arbitrary programs as the user-ID of the web server, allowing them to gain remote access to vulnerable web servers.
10036	IIS ASP source bug	In certain versions of IIS it is possible to read the source to ASP (Active Server Page) files by adding a trailing dot to the URL or by replacing a dot with its hex equivalent. Usually the ASP page will be interpreted on the server to generate the HTML file that a web browser displays.
10037	IIS newdsn.exe bug	The newdsn.exe script that comes with IIS allows users to create databases through a web interface. The script does not check the location of the created database. An attacker can use this script to create or overwrite any file with the permissions of the anonymous internet account (IUSR_machinename). Although the attacker does not control the contents of the created file, it may provide the leverage needed to compromise security, and can easily be used to compromise the availability of a vulnerable server and the machine it runs on.
10038	IRIX MachineInfo Script	Silicon Graphics Irix systems are shipped with a default script in the WWW server cgi-bin directory called MachineInfo. This script allow a remote user to obtain complete information on the system's configuration. Information available includes: 1. Processor type and speed 2. Amount of memory 3. Type of disks installed 4. Type of graphics board
10039	Netscape FastTrack Webserver "get/GET" Bug	Webservers are network servers that speak the HTTP protocol, which is used over TCP connections. One of the commands in the HTTP protocol is "GET", which is used to retrieve HTML files from remote webservers. "GET", like all HTTP commands, must be issued entirely in uppercase; it is a violation of the protocol to use lowercase characters in the command name. Webservers normally issue an error when an HTTP request is malformed. Due to an implementation error, some variants of the Netscape FastTrack webserver do not issue an error, but rather provide a file listing when a "GET" request is issued in lowercase.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
10040	IRIX webdist.cgi check	The webdist.cgi script is shipped with many versions of the Silicon Graphics IRIX operating system. Due to a problem processing CGI arguments, the program incorrectly expands hex-encoded metacharacters without stripping them from the input. The contents of the CGI input to webdist.cgi are passed to the shell when the program executes other commands, so this problem can be used by an attacker to execute arbitrary commands on vulnerable systems.
10042	Microsoft Personal Webserver Overflow DOS	The Microsoft Personal Webserver (MPWS) is a software product that allows workstation users to establish personal web-sites on their desktop machines. Due to a software implementation problem in Microsoft's code, it is possible to cause MPWS to crash by sending an oversized HTTP GET request to the webserver. This can be used to prevent web users from accessing published pages.
10043	IRIX pfdispaly.cgi Vulnerability	The pfdispaly.cgi script is shipped with the Silicon Graphics IRIX operating system as part of the IRIX Performer API Search Tool which is a web based search tool that assists in searching of man pages, documents and example code. The script is loaded by default when installing the IRIX Performer 2.2 CD on IRIX 6.2, 6.3 and 6.4. Due to a problem processing CGI input to pfdispaly the contents are passed to the shell when the program executes other commands, so this problem can be used by an attacker to execute arbitrary commands on vulnerable systems.
10044	FSF "info2www" CGI Check	info2www is a CGI program written in Perl that converts "info"-formatted program documentation into HTML, for viewing over the web via browsers. This script passes an HTTP argument directly to the open() call; an attacker that specifies an argument that includes the pipe character () can thus force the script to execute an arbitrary command.
10046	iCat carbo.dll Vulnerability	The carbo.dll dynamic linked library is shipped with the iCat Carbo Server, a piece Web catalog authoring software for Windows servers. Due to a problem processing user input to carbo.dll the contents are passed to the shell when the program executes other commands, so this problem can be used by an attacker to execute arbitrary commands on vulnerable systems. This vulnerability has been noted in version 3.0.0 of iCat.
10047	"campas" CGI Vulnerability	This module tests for the presence of the "campas" CGI vulnerability on a web server.
10048	HylaFax faxsurvey CGI vulnerability	HylaFax is a package for Unix systems which provides fax services. Included in the packet are web pages for collecting survey information from HylaFax users. The CGI script which is used to gather this information does not properly sanitize the user provided input and evaluates it in a shell.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
10049	WWW faxsurvey check	The faxsurvey CGI program is part of the HylaFAX fax system for UNIX that permits sending and receiving faxes using standard fax capable modems. The faxsurvey program is used to submit forms to a database of HylaFAX users specifying what modems and operating systems they are using the package on. Due to invalid checks on the user's input to the CGI program, it is possible for an attacker to execute arbitrary commands on vulnerable hosts.
10050	Acme's tthttpd - HTTP server GET bug (ver	The command parser in tthttpd removes only the first slash in the filename of GET commands. If you're not running the server in a chrooted environment, an attacker can use this bug to read files outside of your document tree, for instance /etc/passwd.
10053	IIS ism.dll Basic/NTLM Authentication Vulnerability	Versions 2 and 3 of Microsoft Internet Information Web Server (IIS) utilized the ism.dll file for remote administration which was located in the /scripts/iisadmin/ directory. If this dynamic linked library is accessible, remote users may be able to use brute force password guessing techniques to log in and remotely administer the web server. Upgrading IIS from version 2 or 3 to version 4 does not remove this file. If ism.dll is accessible, the forms of remote authentication enabled are returned. Additional information regarding authentication methods is below. IIS allows three forms of authentication for access, anonymous, basic and Windows NT challenge / response (NTLM). Anonymous access requires no password and generally allows remote users to access a public web server. Basic authentication requires a username and password before access is granted. This form of authentication uses base64 encoding which is not encrypted allowing passwords to be sniffed as they pass across the network. NTLM authentication does encrypt usernames and passwords although is generally only available when all hosts use a Microsoft operating system and the web server is not accessed through a proxy server. Furthermore, in certain versions of IIS NTLM authentication is enabled by default. If either form of authentication is enabled username/password combinations can be brute force attacked from remote hosts.
10054	WinGate Logfile Server Vulnerability	WinGate Proxy Server provides a Log File Server on port 8010 to remotely view logfiles. In certain cases this server may be enabled by default. If this service accepts connections from remote hosts, the entire file system may be accessible, allowing remote users to access, read or download any file on your system.
10055	Winroute Administration Port 3129 Vulnerability	Winroute is a Firewall / Proxy Server for Windows that allows remote administration on port 3129. If accessible, remote users can reconfigure sections of the Server, delete all log information and attempt to authenticate as a user.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
10056	IIS Associations reveal webroot Vulnerability	Microsoft's Internet Information Server (IIS) connects all files with programs via file-name extension mapping or associating. The registry key: Hive : HKEY_LOCAL_MACHINE Key : \SYSTEM\CurrentControlSet\Services\W3SVC\Parameters\Script Map shows default associations for the IIS server. A default IIS 3.0 server shows mappings in this registry key such as: .ida -> c:\winnt\system32\idq.dll .idq -> c:\winnt\system32\idq.dll .idc -> c:\winnt\system32\inetsrv\httpodbc.dll By accessing an invalid filename with a valid extension such as "file.idq" in an executable directory the root of the IIS web server maybe revealed.
10057	IIS / ASP Long File Name Denial of Service	Certain versions of Microsoft's Internet Information Server (IIS) Active Server Pages (ASP) are vulnerable to a denial of service attack if they accept file name paths. This module accesses two ASP pages that are part of the default install and attempts to stop the web server from operating.
10064	IIS /scripts Directory Vulnerability	By default, Microsoft's Internet Information Server creates an aliased directory "scripts" which, by default, physically maps to c:\inetpub\scripts. This directory generally contains executable programs or dynamic linked libraries which help perform tasks such as remote administration of the web server. If directory browsing has been enabled on the web server, third parties will be able to remotely browse the scripts directory and identify installed software for use in potential attacks.
10065	Alibaba Web Server ../. Vulnerability	The Alibaba Web Server version 2.0 allows remote attackers to access any files on the same drive as the Alibaba installation. Directory browsing, which is enabled by default, allows the remote browsing of directories on the same drive. If directory browsing is not enabled specific file names can be used for unauthorized access to the web server.
10066	IIS showcode.asp Vulnerability	The Microsoft Internet Information Web Server Version 4.0 contains a number of sample Active Server Page files designed to view the source code of sample applications. One specific file, showcode.asp, does not correctly verify input allowing unauthorized access to files outside the web root of the IIS server.
10067	IIS codebrws.asp Vulnerability	The Microsoft Internet Information Web Server Version 4.0 contains a number of sample Active Server Page files designed to view the source code of sample applications. One specific file, codebrws.asp, does not correctly verify input allowing unauthorized access to files outside the web root of the IIS server.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
10068	Cold Fusion Example Documentation Vulnerability	The Cold Fusion Application Server 4.0 installs by default online documentation allowing remote attackers to view, upload and delete files anywhere on the server. Specifically, Cold Fusion installs the following files: /cfdocs/expeval/openfile.cfm /cfdocs/expeval/displayopenedfile.cfm /cfdocs/expeval/exprcalc.cfm The first file allows remote users to view any file on the server, although as a side effect it also deletes the file viewed. Upon uploading a file to Openfile.cfm it calls displayopenedfile.cfm to display the file which in turn calls exprcalc.cfm to delete the uploaded file. By using exprcalc.cfm to delete itself, remote attackers can then upload arbitrary files to the web server. This module returns a vulnerability if any of the three Cold Fusion examples is accessible.
11000	Network Protocol Spoofing	
11006	RIP spoofing check	The target host was found to be utilizing RIP (Routing Information Protocol) to obtain routing decision information. Version 1 RIP is an easily spoofable protocol. It has been determined that the target host is running RIP version 1.
11010	RST out of TCP window check	A TCP connection between two hosts is identified using the following information: source IP address, source port, destination IP address and destination port. Either end can cause an abrupt close of the connection by sending a TCP packet with the RST flag set, and the correct identifying values. This packet must meet certain criteria in order to be honored by the receiving end. The rules for validating an RST packet that corresponds to an established connection are specified in the Request For Comments 793. Vulnerable hosts do not perform all the validation steps as per RFC 793 and only check for a identifiers that correspond to a valid TCP connection.
11011	IP forwarding check	The target host was found to have IP forwarding enabled.
12000	Packet Filter Verification Tests	
12001	ICMP - echo request	ICMP echo request messages are being forwarded.
12002	ICMP - echo request broadcast	ICMP echo request messages destined for broadcast addresses are being forwarded.
12003	ICMP - echo reply	ICMP echo reply messages are being forwarded.
12004	ICMP - netmask request	ICMP netmask request messages are being forwarded.
12005	ICMP - netmask reply	ICMP netmask reply messages are being forwarded.
12006	ICMP - timestamp request	ICMP timestamp request messages are being forwarded.
12007	ICMP - timestamp reply	ICMP timestamp reply messages are being forwarded.
12008	ICMP - information request	ICMP information request messages are being forwarded
12009	ICMP - information reply	ICMP information reply messages are being forwarded
12010	ICMP - unreachable - network	ICMP network unreachable messages are being forwarded.
12011	ICMP - unreachable - host	ICMP host unreachable messages are being forwarded.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
12012	ICMP - unreachable - protocol	ICMP protocol unreachable message.
12013	ICMP - unreachable - port	ICMP port unreachable message.
12014	ICMP - fragmentation needed but DF set	ICMP unreachable, fragmentation needed but don't-fragment bit set messages forwarded.
12015	ICMP - source route failed	ICMP unreachable - source route failed messages are being forwarded
12016	ICMP - destination network unknown	ICMP unreachable - destination network unknown messages are being forwarded
12017	ICMP - destination host unknown	ICMP unreachable - destination host unknown messages are being forwarded
12018	ICMP - source host isolated	ICMP unreachable - source host isolated messages are being forwarded.
12019	ICMP - destination network administratively prohibited	ICMP unreachable - destination network administratively prohibited.
12020	ICMP - destination host administratively prohibited	ICMP unreachable - destination host administratively prohibited.
12021	ICMP - network unreachable for TOS	ICMP - network unreachable for TOS messages are being forwarded.
12022	ICMP - host unreachable for TOS	ICMP - host unreachable for TOS messages are being forwarded.
12023	ICMP unreachable - communication administratively prohibited	ICMP unreachable - communication administratively prohibited messages are being forwarded.
12024	ICMP unreachable - host precedence violation	ICMP unreachable - host precedence violation messages are being forwarded
12025	ICMP unreachable - precedence cutoff in effect	ICMP unreachable - precedence cutoff in effect messages are being forwarded
12026	ICMP - source quench	ICMP source quench messages are being forwarded.
12027	ICMP - redirect - network	ICMP network redirect packets are being forwarded.
12028	ICMP - redirect - host	ICMP host redirect packets are being forwarded.
12029	ICMP - redirect - TOS and network	ICMP TOS and network redirect packets are being forwarded.
12030	ICMP - redirect - TOS and host	ICMP TOS and host redirect packets are being forwarded.
12031	ICMP - router advertisement	ICMP router advertisement messages are being forwarded.
12032	ICMP - router solicitation	ICMP router solicitation messages are being forwarded.
12033	ICMP - TTL expired in transit	ICMP - TTL expired in transit messages are being forwarded
12034	ICMP - TTL expired during assembly	ICMP - TTL expired during assembly messages are being forwarded
12035	ICMP - IP header bad	ICMP - IP header bad messages are being forwarded
12036	ICMP - required option missing	ICMP - required option missing messages are being forwarded

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
12040	IP - fragmentation - tiny fragment	The firewall was found to forward packets with an initial fragment size smaller than 76 bytes. The forwarding of packets smaller than 76 bytes can allow an attacker to evade packet filters which filter based on the location and values of TCP flags within the TCP header. A common filtering technique is to prevent new incoming TCP connections from being established through a firewall by looking for the SYN flag within the TCP header. If this flag is set, the connection is denied, if coming from the outside. This vulnerability evades the packet filter by placing the flags portion of the TCP packet into the next packet. The following excerpt is taken from RFC 1858: With many IP implementations it is possible to impose an unusually small fragment size on outgoing packets. If the fragment size is made small enough to force some of a TCP packet's TCP header fields into the second fragment, filter rules that specify patterns for those fields will not match. If the filtering implementation does not enforce a minimum fragment size, a disallowed packet might be passed because it didn't hit a match in the filter.
12041	IP - fragmentation - tiny fragment without MF bit set	The firewall was found to forward packets with a fragment size smaller than 76 bytes. The forwarding of packets smaller than 76 bytes can allow an attacker to evade packet filters which filter based on the location and values of TCP flags within the TCP header. A common filtering technique is to prevent new incoming TCP connections from being established through a firewall by looking for the SYN flag within the TCP header. If this flag is set, the connection is denied, if coming from the outside. This vulnerability evades the packet filter by placing the flags portion of the TCP packet into the next packet. The following excerpt is taken from RFC 1858: With many IP implementations it is possible to impose an unusually small fragment size on outgoing packets. If the fragment size is made small enough to force some of a TCP packet's TCP header fields into the second fragment, filter rules that specify patterns for those fields will not match. If the filtering implementation does not enforce a minimum fragment size, a disallowed packet might be passed because it didn't hit a match in the filter.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
12042	IP - fragmentation - tiny fragment with reserved bit set	The firewall was found to forward tiny fragmented IP datagrams which have the MF (More Fragments) flag set and the reserved flag set. The forwarding of packets smaller than 76 bytes can allow an attacker to evade packet filters which filter based on the location and values of TCP flags within the TCP header. A common filtering technique is to prevent new incoming TCP connections from being established through a firewall by looking for the SYN flag within the TCP header. If this flag is set, the connection is denied, if coming from the outside. This vulnerability evades the packet filter by placing the flags portion of the TCP packet into the next packet. The following excerpt is taken from RFC 1858: With many IP implementations it is possible to impose an unusually small fragment size on outgoing packets. If the fragment size is made small enough to force some of a TCP packet's TCP header fields into the second fragment, filter rules that specify patterns for those fields will not match. If the filtering implementation does not enforce a minimum fragment size, a disallowed packet might be passed because it didn't hit a match in the filter.
12043	IP - fragmentation - second fragment has offset of 1	The firewall was found to forward fragmented IP datagrams which contain a fragment offset of 1. The forwarding of fragments with an offset that overlaps other fragments may allow an attacker to evade packet filters by overwriting header values in the initial fragment with values that would not have normally been permitted. The outcome of overlapping IP fragments depends on whether the destination IP stack prefers old or new data when reassembling IP fragments.
12044	IP - options - strict source route	IP datagrams containing strict source route options are being forwarded
12045	IP - options - loose source route	IP datagrams containing loose source route options are being forwarded
12046	IP - options - record route	IP datagrams containing record route options are being forwarded
12047	IP - options - timestamp	IP datagrams containing timestamp options are being forwarded
12048	IP - version - less than 4	IP datagrams with an IP version number of less than 4 are being forwarded.
12049	IP - version - greater than 4	IP datagrams with an IP version number of greater than 4 are being forwarded.
12050	IP - TCP protocol permitted	TCP packets are being forwarded by the firewall without significant filtering.
12051	IP - UDP protocol permitted	UDP packets are being forwarded by the firewall without significant filtering.
12052	IP - odd protocols	IP datagrams containing an uncommon protocol are being forwarded by the firewall.
12060	TCP - common ports permitted	One or more common TCP services are being forwarded by the firewall.
12061	TCP - ports permitted [exhaustive]	One or more TCP services are being forwarded by the firewall.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
12062	TCP - source ports permitted	The firewall allows connections originating from one or more common source ports into the internal network.
12070	UDP - common ports permitted	One or more common services are being forwarded by the firewall.
12071	UDP - ports permitted [exhaustive]	One or more common services are being forwarded by the firewall.
12072	UDP - source ports permitted	The firewall allows connections originating from one or more common source ports into the internal network.
13000	Firewalls, Filters, and Proxies	
13001	Livingston Portmaster fixed TCP ISN check	This module checks if a Livingston Portmaster router is vulnerable to TCP sequence prediction attacks. A router that is vulnerable to this attack is open to spoofing and TCP session hijacking attacks where the intruder can take over an established session and gain complete control of the router's configuration. Livingston Portmaster routers are particularly vulnerable since they use the same fixed TCP initial sequence number for all TCP sessions.
13002	TCP sequence numbers are predictable	The target host was found to be vulnerable to TCP sequence number prediction attacks. The host generates TCP sequence numbers in a pattern which can be guessed by an intruder to launch TCP spoofing based attacks.
13005	SOCKS version 4 configuration check	This check attempts to access services through an incorrectly configured SOCKS version 4 proxy. A connection was established through the proxy server back to the scanning host.
13011	Wingate POP3 proxy Username Overflow check	Wingate POP3 proxy Username Overflow check This module determines whether the remote POP3 server is vulnerable to a buffer overflow attack when parsing the user login name. By providing the daemon with a long username, an attacker can overflow the username buffer and cause the server to crash. It may be possible for an attacker to cause the server to run arbitrary programs by providing a carefully crafted username. A vulnerable Wingate proxy will stop responding to legitimate clients after the attack is performed. Notice: Certain versions of SCO Unix ship with a POP3 service enabled that is vulnerable to a similar serious problem, in which an attacker can exploit a buffer overflow triggered by any overly-large command. Because the test for this specific POP3 vulnerability involves the transmission of an extremely large POP command, this test may flag vulnerable SCO POP servers as well.
13012	IGMP host poll check	This check attempts to gather a list of hostnames from routers which support Multicasting groups.
13013	Unpassworded WinGate Proxy Server	WinGate is a proxy server for Windows environments. It allows multiple machines to share a single connection and IP address by proxying all requests through a single server. An unpassworded WinGate server can be used to launder connections for unauthorized and illegal network usage. WinGate is exploited by connecting to the "telnet" port of the proxy server, and using the command-line interface to create a new outbound connection to an arbitrary address. This new connection can be used to attack other hosts.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
14000	Authentication Mechanisms	
14001	NIS+ Incorrect permissions on passwd.org_dir table	The permissions on the passwd.org_dir table were found to be set incorrectly. In many cases the permissions on the default NIS+ installation are set incorrectly. This may allow unauthorized access to table information.
14002	NIS+ Incorrect permissions on passwd.org_dir columns	The permissions on the specific columns within the passwd.org_dir table were found to be set incorrectly. In many cases the permissions on the default NIS+ installation are set incorrectly. This may allow unauthorized access to table information.
14003	NIS+ Incorrect permissions on passwd.org_dir entries	The permissions on the specific entries within the passwd.org_dir table were found to be set incorrectly.
14004	NIS+ Security level retrieval	This module prints out the security level which the NIS+ server on the target host is currently running at. NIS+ supports 3 different levels of security: Level 0 : No access control whatsoever is performed Level 1 : AUTH_SYS credentials are allowed, AUTH_SYS credentials are easily forged by users and should not be used. Level 2 : Only AUTH_DES credentials are accepted. This should be the security level for normal operation.
14005	NIS+ Dangerous security level	This module determines whether the target NIS+ server is running at a security level below 2. If the NIS+ server is running at any security level lower than 2, attackers can trivially modify and retrieve NIS+ information.
14006	NIS+ Process ID gathering	This module utilizes a feature of the NIS+ server, which allows remote users to determine whether a particular process ID is running on the target server.
14007	NIS+ rpc.nisd remote buffer overflow	The target host was found to be vulnerable to a buffer overflow vulnerability in the rpc.nisd RPC service. This service is present on workstations and servers running the Sun Microsystems Solaris operating system and utilizing the NIS+ suite. By sending data consisting of an abnormally long text string within a valid NIS+ RPC packet, an overflow within the NIS+ server occurs. By sending correctly formed data, an attacker can exploit this buffer overflow to run commands on the target system. WARNING: If enabled, this module will crash a vulnerable NIS+ server. If this module returns positive, ensure that you are prepared to restart this service.
15000	General Remote Services	
15001	Open X Server check	The X Windows server running on the target host was found to allow unrestricted access. Some operating systems are shipped without any access restrictions to the X Windows server.
15003	Xterm cookie guess check	Some versions of X windows use MIT style magic cookies for authentication. However in some version of X these cookies are guessable, making your Xterm open to attack as if it had no access control whatsoever.
15004	Telnet LD_LIBRARY_PATH vulnerability	The telnet daemon on the target host was found to be vulnerable to a security problem which may allow an attacker to obtain remote super-user access to the system.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
15005	POP shadowed password vulnerability	The target host was found to be running a vulnerable version of the POP3 server for Linux. A known vulnerability in older Linux installations which also have the shadow password suite installed allowed an attacker to read any user's mail via the POP3 service.
15006	rlogin -froot check	On some versions of AIX and Linux a remote user can gain root access by exploiting a problem in rlogind. This problem is a result of incorrectly parsing the parameters passed to the login program, which results in the attacker having the ability to login as the root user, without a password.
15007	Kerberos server c check	This check discerns whether a target Kerberos server (V4) can be coaxed into offering up valid ciphered passwords. Passwords encrypted under Kerberos (V4) can be decrypted much in the same way UNIX password files can.
15008	UUCP service check	This module discerns whether the UUCP service is offered on a host. Many network connected systems are shipped with the UUCP service enabled by default. This may open up potential security problems.
15009	Open news server check	This module checks to see if it can read or post news articles off your News Server. If this is possible, a remote user can poll your news feed causing a strain on your system resources. Moreover they can post erroneous information from your news server which may be embarrassing to your company image.
15011	cfingerd (1) exploit check	This module attempts to exploit a vulnerability in earlier versions of cfingerd for Linux, which could lead to root compromise. This bug is related to cfingerd parsing instructions from incoming fingers incorrectly.
15014	Telnet RESOLV_HOST_CONF check	Some telnet daemons will accept environment variables from remote telnet clients. Some of these variables include paths to system files. A vulnerability exists in some systems' resolver library whereby a user can specify the location of a configuration file. If your host is vulnerable to this, an intruder could read any file on your system by connecting to your telnet daemon.
15015	Radiusd overflow check	Some versions of radiusd have a weakness whereby a buffer overflow can be exploited to gain a segfault in the daemon and perhaps execute arbitrary commands as root.
15020	Linux NIS+ account	In the past installations of NIS+ on some Linux distributions were configured improperly in the /etc/passwd file. This inconsistency allowed for remote users to log in as '+'.
15021	Hosts.equiv (+) check	This module check's if your hosts.equiv is misconfigured with a '+' in it which would allow for users to rsh (or any other 'r' service for that matter) into your host.
15024	HP Remote Watch check	This module determines whether your HP-UX system is vulnerable to a bug in the HP Remote Watch package whereby a remote user can easily obtain root access on your host.
15025	Kerberos user name gathering check	This check attempts to coax usernames and the Kerberos realm from a Kerberos server. This allows users to match up usernames with a list of gathered ciphered passwords which they could crack.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
15026	Linux TFTP (Trivial File Transfer Protocol) check	This module checks for a faulty access control implementation in Linux versions of the tftpd daemon. Most current tftpd implementations attempt to restrict access to files outside of the tftproot directory. The Linux implementations disallow any files with ../ in their pathnames, however one can still access files such as /etc/passwd by prepending ../ in front of the pathname (../etc/passwd). This will work since the current directory for tftpd is usually /ftpchroot.
15027	IMAP and POP buffer overflow check	Several versions of both IMAP and POP servers which provide remote mail management contain a serious vulnerability. This check determines whether your IMAP daemon is vulnerable to a buffer overflow which allows users to execute arbitrary commands on your server. This vulnerability allows users to execute commands remotely as root.
15028	INN control message check	This check determines whether your version of INN is vulnerable to a problem which allows remote users to execute commands on your news server. This can be done by feeding your news server control messages with shell escape characters in them, causing INN to execute commands. This test attempts to determine your INN version number. INN versions earlier than 1.5.1 have a number of problems with their parsing of control messages, resulting in information from message headers being passed to a shell.
15029	INN nnrpd buffer overflow	This check determines whether your news server is vulnerable to a buffer overflow present in the nnrpd program. The nnrpd program is run by the INN news server software to handle the reading and posting of usenet articles by users. A vulnerability in this program can allow remote users to execute arbitrary commands on your news server.
15030	SSH Version 1.2.17 check	Version 1.2.17 of the SSH server package contains security vulnerabilities which can lead to an attacker compromising the security of the SSH protocol. This vulnerability is present in version 1.5 of the SSH protocol which is only present in version 1.2.17 of the SSH package.
15031	Vacation remote execution vulnerability	Vacation is used by the recipient of email messages to notify the sender that they are not currently reading their mail. A vulnerability exists within the vacation program which allows individuals to execute commands remotely.
15032	Perl fingerd 0.2	Version 0.2 of the perl fingerd passes remote usernames to a shell. Thus, passing the fingerd a username containing shell metacharacters can cause it to execute arbitrary commands remotely.
15033	DG/UX fingerd	Some versions of the DG/UX fingerd pass their input to a shell. This makes it possible for remote attackers to execute arbitrary commands on the DG/UX system.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
15034	Telnet Daemon TERMCAP check	This module determines whether the remote telnet daemon is vulnerable to a buffer overflow attack when parsing a terminal capability file. By uploading an alternate termcap file, an attacker can specify the path to this file and cause the telnet daemon to execute arbitrary commands.
15035	POP3 Username Overflow check	This module determines whether the remote POP3 server is vulnerable to a buffer overflow attack when parsing the user login name. By providing the daemon with a long username, an attacker can overflow the username buffer and cause the server to crash. It may be possible for an attacker to cause the server to run arbitrary programs by providing a carefully crafted username. If the POP3 server is the Seattle Lab Mail Server package, crashing the POP3 server causes the entire mail server to stop. Notice: Certain versions of SCO Unix ship with a POP3 service enabled that is vulnerable to a similar serious problem, in which an attacker can exploit a buffer overflow triggered by any overly-large command. Because the test for this specific POP3 vulnerability involves the transmission of an extremely large POP command, this test may flag vulnerable SCO POP servers as well.
15036	SCO POP Overflow check	This module determines whether the remote POP server is vulnerable to a buffer overflow attack. Santa Cruz Operation OpenServer 5.0.0 through 5.0.4, Internet FastStart 1.0.0 and 1.1.0 are known to be vulnerable to this attack. By providing the daemon with a long string, an attacker can overflow an internal buffer and cause the server to execute arbitrary commands as root.
15037	Null Rsh Check	This module determines whether a remote user is able to login to the target system by specifying a NULL username. The in.rshd daemon on some systems would allow logins from NULL users due to a vulnerability in the ruserok() library call.
15038	Solaris in.rlogind FTP bounce vulnerability	This module determines whether the rlogin daemon on the target host is vulnerable to an FTP bounce attack. This vulnerability relies on the ability of an attacker to subvert the FTP daemon on the target host to connect to the rlogin service port on the target host, and execute arbitrary commands. This module determines whether the target server's rlogin daemon is vulnerable to this attack. In order to be exploited however, the FTP daemon must also be running on the target host. This module does not determine whether the FTP server is running. While this may not be an exploitable vulnerability at this time, it is possible that an FTP server may be running on the target host in the future.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
15039	Qualcomm "qpopper" POP3 command vulnerability	Some versions of the the Qualcomm "qpopper" POP3 service contain a vulnerability which allows an attacker to execute arbitrary commands remotely as the super-user. This module checks to see if this vulnerability is present. Notice: Certain versions of SCO Unix ship with a POP3 service enabled that is vulnerable to a similar serious problem, in which an attacker can exploit a buffer overflow triggered by any overly -large command. Because the test for this specific POP3 vulnerability involves the transmission of an extremely large POP command, this test may flag vulnerable SCO POP servers as well.
15040	Qualcomm "qpopper" POP3 PASS Overflow	Some versions of the the Qualcomm "qpopper" POP3 service contain a vulnerability which allows an attacker to execute arbitrary commands remotely as the super-user. This module checks to see if this vulnerability is present. Notice: Certain versions of SCO Unix ship with a POP3 service enabled that is vulnerable to a similar serious problem, in which an attacker can exploit a buffer overflow triggered by any overly -large command. Because the test for this specific POP3 vulnerability involves the transmission of an extremely large POP command, this test may flag vulnerable SCO POP servers as well.
15043	TFTP (Trivial File Transfer Protocol) readable	The TFTP service running on the target host was found to allow the retrieval of arbitrary files.
15044	TFTP (Trivial File Transfer Protocol) writable	The TFTP service running on the target host was found to allow arbitrary files to be created and written to anywhere on the target system.
15045	SSH RhostsAuthentication enabled	The SSH service running on the target host was found to have rhosts authentication enabled. rhosts authentication provides access verification based on the source address of the client user, and is susceptible to IP address spoofing, and DNS cache corruption attacks.
15047	BNC IRC Proxy Remote Overflow	BNC is an IRC proxy package that allows IRC chat clients to obtain forwarded access to IRC servers. BNC listens on a user- configurable port for connections, and forwards them to an IRC server. Due to an implementation problem inside the proxy server, it is possible for a remote attacker to gain access to the shell account the BNC proxy is running under. This attack, which exploits a buffer overflow in the proxy's command processing code, effectively allows an attacker complete access to the machine the proxy server is running on.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
15048	CSM Proxy 4.1 Denial of service	This check determines whether you can crash the CSM Proxy 4.1 by sending 1030 characters or more to the FTP port (21). The CSM Proxy accepts connections and username/password information before checking for authorization based on source IP address of the connection. This allows any user on the Internet/Intranet to connect to the proxy server (even from an unauthorized host) and exploit a buffer overflow problem that makes the CSM Proxy crash (in addition to the Windows NT machine that it is running on) when it receives more than 1029 characters in its FTP port (port 21/tcp). If the CSM Proxy is running on a host protected by a firewall and not accessible from the Internet, this vulnerability can only be exploited by users on hosts of the internal network. Notice: Under certain circumstances the UNIX version of the vulnerable CSM Proxy may not crash, although its memory usage will significantly increase.
16000	SMB/NetBIOS Resource Sharing	
16001	Unpassworded NetBIOS/SMB check	Service Message Block (SMB) is the standard resource-sharing protocol used by Windows platforms. The SMB protocol is transmitted using NetBIOS, a networking protocol designed to allow groups of PCs to interoperate. NetBIOS is accessible over TCP/IP using the NBT protocol. SMB resource sharing makes use of two different security models, "share-level" and "user-level". In share-level security, groups of files (directory trees) are protected by a password, allowing simple workgroups to be configured simply by ensuring that they share a password. In user-level security, all attempts to access resources are authenticated with a username and password. It is possible to obtain a list of shares offered by an SMB-speaking computer by initiating an SMB session with no username or password (this is referred to as a "null session"). The information available from this transaction can be used by an attacker to conduct further attacks.
16002	Guessable NetBIOS/SMB password check	Service Message Block (SMB) is the standard resource-sharing protocol used by Windows platforms. The SMB protocol is transmitted using NetBIOS, a networking protocol designed to allow groups of PCs to interoperate. NetBIOS is accessible over TCP/IP using the NBT protocol. SMB resource sharing makes use of two different security models, "share-level" and "user-level". In share-level security, groups of files (directory trees) are protected by a password, allowing simple workgroups to be configured simply by ensuring that they share a password. In user-level security, all attempts to access resources are authenticated with a username and password. This check attempts to connect to the remote NetBIOS file sharing service and attempt to login with common passwords and accounts which are enabled with Windows NT by default. If successful, this will allow an unauthorized user to access shares and services which are being offered by the remote host. (Note: the usernames and passwords used are not taken from the userlist or password list files).

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
16003	SMB LANMAN Pipe Server information gathering	Service Message Block (SMB) is the standard resource-sharing protocol used by Windows platforms. The SMB protocol is transmitted using NetBIOS, a networking protocol designed to allow groups of PCs to interoperate. NetBIOS is accessible over TCP/IP using the NBT protocol. One resource SMB servers make available to clients is an IPC mechanism called "transaction pipes". A transaction pipe allows SMB clients to communicate with remote servers using the SMB protocol as a transport. Transaction pipes are accessed via special "file names" from SMB hosts. Among the transaction pipes available to clients of Windows NT servers is "\\PIPE\LANMAN", over which the Remote Administration Protocol (RAP) is spoken. Using the LANMAN pipe, it is possible to collect a great deal of information about the configuration and status of an NT server. Information available from the LANMAN pipe includes version and vendor information, along with NT server, workgroup, and domain names. This information can be useful to an attacker when looking for weaknesses in particular server implementations.
16004	SMB LANMAN Pipe Share listing	Service Message Block (SMB) is the standard resource-sharing protocol used by Windows platforms. The SMB protocol is transmitted using NetBIOS, a networking protocol designed to allow groups of PCs to interoperate. NetBIOS is accessible over TCP/IP using the NBT protocol. One resource SMB servers make available to clients is an IPC mechanism called "transaction pipes". A transaction pipe allows SMB clients to communicate with remote servers using the SMB protocol as a transport. Transaction pipes are accessed via special "file names" from SMB hosts. Among the transaction pipes available to clients of Windows NT servers is "\\PIPE\LANMAN", over which the Remote Administration Protocol (RAP) is spoken. Using the LANMAN pipe, it is possible to collect a great deal of information about the configuration and status of an NT server. Information available from the LANMAN pipe includes a list of shares available on the NT server. This provides an attacker a listing of directories and file systems which are being offered, giving an attacker a target filesystem or service to attempt to abuse.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
16005	SMB LANMAN Pipe Server browse listing	Service Message Block (SMB) is the standard resource-sharing protocol used by Windows platforms. The SMB protocol is transmitted using NetBIOS, a networking protocol designed to allow groups of PCs to interoperate. NetBIOS is accessible over TCP/IP using the NBT protocol. One resource SMB servers make available to clients is an IPC mechanism called "transaction pipes". A transaction pipe allows SMB clients to communicate with remote servers using the SMB protocol as a transport. Transaction pipes are accessed via special "file names" from SMB hosts. Among the transaction pipes available to clients of Windows NT servers is "\\PIPE\LANMAN", over which the Remote Administration Protocol (RAP) is spoken. Using the LANMAN pipe, it is possible to collect a great deal of information about the configuration and status of an NT server. The information available from an NT server via the LANMAN pipe includes the "browse listing" of the system, which lists the names of other SMB-speaking systems that the server communicates. This information can provide an attacker with an easy way to obtain new target systems to attack.
16006	NetBIOS/SMB Accessible Share	Service Message Block (SMB) is the standard resource-sharing protocol used by Windows platforms. The SMB protocol is transmitted using NetBIOS, a networking protocol designed to allow groups of PCs to interoperate. NetBIOS is accessible over TCP/IP using the NBT protocol. SMB resource sharing makes use of two different security models, "share-level" and "user-level". In share-level security, groups of files (directory trees) are protected by a password, allowing simple workgroups to be configured simply by ensuring that they share a password. In user-level security, all attempts to access resources are authenticated with a username and password. By manipulating the SMB protocol and services offered by Windows NT, it is possible to obtain a list of shares exported by an SMB service. In addition, Windows SMB servers tend to have several common shares available, the presence of which can be guessed without attempting to obtain a share list. This check attempts to access all shares which are being served by the remote server. If any shares are accessible, an intruder can possibly read or write data from and to the share. This can lead to data being stolen, or modified on the server.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
16007	NetBIOS/SMB Hidden Share	Service Message Block (SMB) is the standard resource-sharing protocol used by Windows platforms. The SMB protocol is transmitted using NetBIOS, a networking protocol designed to allow groups of PCs to interoperate. NetBIOS is accessible over TCP/IP using the NBT protocol. SMB resource sharing makes use of two different security models, "share-level" and "user-level". In share-level security, groups of files (directory trees) are protected by a password, allowing simple workgroups to be configured simply by ensuring that they share a password. In user-level security, all attempts to access resources are authenticated with a username and password. Although it is possible, by manipulating the SMB protocol and services offered by Windows NT, to obtain a list of shares, many SMB servers also have several common share names available, including the "ROOT" share and the root directory of MS-DOS hard drive partitions. An attacker can guess the names of these shares and verify their presence using the SMB protocol, and thus gain information that can be used to launch further attacks against the system. An attacker that can gain access to these shares can potentially read or modify the data they contain.
16008	NetBIOS/SMB Writable Share Check	Service Message Block (SMB) is the standard resource-sharing protocol used by Windows platforms. The SMB protocol is transmitted using NetBIOS, a networking protocol designed to allow groups of PCs to interoperate. NetBIOS is accessible over TCP/IP using the NBT protocol. SMB resource sharing makes use of two different security models, "share-level" and "user-level". In share-level security, groups of files (directory trees) are protected by a password, allowing simple workgroups to be configured simply by ensuring that they share a password. In user-level security, all attempts to access resources are authenticated with a username and password. This check confirms that a share which has been determined to be accessible to an attacker is also writable. An attacker with write access to a share can modify the data it contains, violating the integrity of that data and potentially the entire system.

JOHANNESBURG

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
16009	NetBIOS/SMB Dot Dot Bug	Service Message Block (SMB) is the standard resource-sharing protocol used by Windows platforms. The SMB protocol is transmitted using NetBIOS, a networking protocol designed to allow groups of PCs to interoperate. NetBIOS is accessible over TCP/IP using the NBT protocol. SMB resource sharing makes use of two different security models, "share-level" and "user-level". In share-level security, groups of files (directory trees) are protected by a password, allowing simple workgroups to be configured simply by ensuring that they share a password. In user-level security, all attempts to access resources are authenticated with a username and password. SMB shares specify collections of files that are accessible to an SMB client. Data outside the specified SMB share on the server should not be accessible to a client; this allows selective portions of a filesystem to be shared via SMB. Complete access to the filesystem of an SMB server would allow clients to access and modify it's configuration, thus compromising the integrity of the system. In some SMB implementations, permutations of the "." directory are handled incorrectly, allowing an attacker to access data outside the exported share. This check attempts to circumvent directory protection by exercising this bug.
16020	NetBIOS Name Table Retrieval	This check obtains the system name tables from the remote system's NetBIOS name service.
16021	NetBIOS Name Table Registration	This module performs a NetBIOS name registration to register a false machine name on the target host.
16022	NetBIOS Name Table De-registration	This module performs a NetBIOS name release to de-register NetBIOS name table entries.
16023	NetBIOS Samba login defaults to GUEST	Samba is a NetBIOS/SMB file sharing package available for Unix based operating systems, allowing interoperability with Windows NT file sharing. The Samba server found on the target host has been found to default to a GUEST login, if a valid username and password are not entered.
16024	NetBIOS Samba password buffer overflow	The Samba NetBIOS distribution on the target host contains a buffer overflow vulnerability which can allow remote users to execute arbitrary commands on the server. By specifying a correctly formatted password string that is longer than what Samba is expecting, a buffer overflow occurs. Versions of Samba prior to 1.9.17p2 are vulnerable to this attack.
17000	Domain Name System and BIND	
17002	DNS Supports IQUERY check	This module determines whether or not the remote nameserver supports the IQUERY operation. The IQUERY function in named implementations is fed an IP range (netmask) and will return all available resource records for the hosts within the given range.
17004	DNS Zone transfer check	This module determines whether or not zone transfers are supported by the given nameserver.
17005	DNS Zone transfer by exhaustive search using IQUERY	If the specified nameserver does not allow zone transfers, it is still possible in most cases to obtain the same information, and resource records by iteratively using the IQUERY operation to build a listing of the domain.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
17007	DNS Server allows Updates	This checks if the target DNS was compiled with the '-DALLOW_UPDATES' option. '-DALLOW_UPDATES' is an extension which allows for dynamic updating of name service information. The dynamic update code in BIND, as noted by its author Mike Schwartz (schwartz@cs.washington.edu), ignores all security issues. As a result, any DNS compiled with -DALLOW_UPDATES can be easily fooled into changing resource records of the zones it serves. These updates will also be propagated to secondary name servers.
17008	DNS additional info piggybacked in a QUERY check	This module determines whether or not a host will cache information which is appended to the end of a legitimate query. It is highly unlikely that current implementations support this, however this was supported in old BIND implementations. We query the server for a legitimate host, and add additional resource records to the back of the query. Then we determine whether the server has cached this additional record or not.
17010	DNS accepts responses out of sequence check	This module determines whether a DNS server will accept responses with invalid ID numbers. We query the DNS server for a host which is resolved somewhere else on the Internet, and send a fake reply with a false ID number. If our response is cached, we conclude that the server is caching responses with invalid ID numbers.
17014	DNS caches answers with binary data check	Determine whether or not the DNS server will cache binary data in hostname queries. Caching binary data in place of hostname information is very dangerous as many programs expect the nameserver to return clean, valid printable information. It has been noted that many programs can be exploited by passing invalid data via DNS responses. We query the nameserver for a legitimate host, and respond with a legitimate reply containing invalid binary data. We then query the DNS server again to determine if this was cached or not. For reference: BIND 4.8.3 allows caching anything you want. BIND 4.9.3 will cache under certain conditions. BIND 4.9.4-P1 will not cache binary data
17018	DNS version number check	This module attempts to obtain the remote version number from the DNS server. This information is provided by post 4.9.5 BIND name servers. The information consists of the version of BIND running on the remote server, and the host and user who compiled the installed nameserver.
17020	DNS Cache Corruption, Guessable Query IDs	Most nameservers on the Internet are vulnerable to an attack that allows an attacker to cache arbitrary information on the server, thus allowing the attacker to spoof DNS, redirect web traffic, and subvert hostname-based authentication. This attack works by forcing the target nameserver to attempt to resolve the information being spoofed, and then forging the response to this request. To do this, the attacker needs to be able to predict the query-ID used by the target nameserver in the query. This module attempts to determine whether or not the target nameserver uses query IDs which can be predicted. If it is determined that the query IDs are predictable, an attacker can forge responses to DNS queries and spoof the DNS protocol.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
17021	DNS Cache Corruption, Multiple-Answer Attack	Recent revisions of BIND (4.9.5 and below) are vulnerable to an attack that allows arbitrary individuals on the network to cache incorrect information on the server. This allows an attacker to spoof nameservice, redirect web accesses, and bypass name-based authentication (such as TCP-wrappers). The attack involves forcing the nameserver to talk to another server somewhere else on the network, in order to resolve some random name. The remote server responds to this query with two answers, one answering the query, and another that contains false information. Vulnerable servers will cache both answers, and the fake data will be made available for future queries.
17022	DNS Cache Corruption, Poisoned-NS Attack	Recent revisions of BIND (4.9.5 and below) are vulnerable to an attack that allows arbitrary individuals on the network to cache incorrect information on the server. This allows an attacker to spoof nameservice, redirect web accesses, and bypass name-based authentication (such as TCP-wrappers). This attack works by forcing the nameserver to talk to a remote server to resolve a query for some random name. The remote server can trick the nameserver into caching arbitrary names by responding to this query with an answer that contains a fake NS record; the information from this NS record will be cached on the target nameserver.
17023	DNS Cache Corruption, Parallel Query Attack	Most nameservers on the Internet are vulnerable to an attack that allows an attacker to cache arbitrary information on the server, thus allowing the attacker to spoof DNS, redirect web traffic, and subvert hostname-based authentication. This attack works by forcing the target nameserver to attempt to resolve the information being spoofed, and then forging the response to this request. To do this, the attacker needs to be able to predict the query-ID used by the target nameserver in the query. The effectiveness of this attack can be heightened by forcing the target nameserver to launch many queries for this information in parallel, thus causing it to allocate more query IDs, which gives an attacker a greater opportunity to guess the query ID, even if it's randomized. This module attempts to determine if an attacker can force the nameserver to initiate multiple queries for the exact same information. If the nameserver does this, an attacker can significantly increase the odds of successfully guessing query IDs and forging DNS responses.
17024	DNS IQUERY Buffer Overflow Attack	Certain versions of BIND are vulnerable to an attack which allows a remote DNS client to run an arbitrary command on the nameserver host as the user the server runs as (frequently root). This attack exploits an implementation flaw in BIND that involves a buffer overflow triggered by inserting an overly long name record into a DNS IQUERY request. Most BIND servers do not support the IQUERY operation. These servers are not vulnerable to this attack. However, many Linux hosts run stock nameservers which are configured to support IQUERY; these hosts can be compromised completely by this attack.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
18000	Windows NT - Network Vulnerabilities	
18001	Connection to IPC\$ as Anonymous User Allowed	The remote host allows the Anonymous user to establish connections to the IPC\$ share over the network. The IPC\$ share is used by Windows NT to provide a number of system administration services to other networked users. Unix machines running the Samba SMB service also make an IPC\$ share available over the network.
18002	Password Grinding (through IPC\$)	Users may remotely use the services of an NT machine by connecting to one of the shares. In order to connect to a share the user must provide an account name and a password. This module attempts to connect to the IPC\$ share (used for remote communication with system services) by trying a number of users and passwords. If a username and password is guessed, it may be used to get protected information, connect to other shares or even log in to the machine.
18003	Registry permission problems	This module looks through the remotely accessible parts of the registry looking for permissions that allow remote users to modify the registry without an account on the system. In general remote users should not be allowed to change the configuration information of the machine without an account. The impact of having permission problems can range from benign, to allowing denial of service attacks, to allowing compromise of the systems accounts.
18004	Password Database Retrieved	This module grabs the password database from a remote NT machine. This module does not demonstrate a vulnerability but rather grabs extra information that would be available to an attacker who has compromised the Administrator account.
18005	LSA Secrets Retrieved	This module grabs the Services secrets stored in the Local Security Authority. This module does not demonstrate a vulnerability but rather obtains extra information that would be available to an attacker who has compromised the Administrator account.
18007	Lan Manager Authentication Enabled	The target host was found to have Lan Manager authentication enabled. Lan Manager authentication is a weaker form of authentication which can be easily cracked by an attacker. Your security policy indicates that Lan Manager authentication should be disabled.
18008	Force server to use SMB message signing	The security policy indicates that servers must use SMB message signing on all SMB traffic. The host is currently not configured to do so. SMB message signing causes each packet to be signed by the sender, allowing verification by both the client and server end, ensuring that no data has been tampered with by an attacker.
18009	Force client to use SMB message signing	The security policy indicates that clients must use SMB message signing on all SMB traffic. The host is currently not configured to do so. SMB message signing causes each packet to be signed by the sender, allowing verification by both the client and server end, ensuring that no data has been tampered with by an attacker.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
18010	Registry Access Not Restricted	The restrictions on the Windows NT Registry were found to allow access to all users. Access to the registry via the network is governed by the restrictions imposed on the "winreg" registry key. This key was found either to be missing or to contain permissions allowing access to "Everyone".
18011	DCOM Support Enabled (remote activation of COM servers)	The target host has been found to have DCOM enabled. This controls the global activation and call policies of the machine
18012	DCOMRunAs Value Writable	The target host has been found to have the DCOM RunAs value writable by the Interactive user.
18013	Registry HKEY_LOCAL_MACHINE Key writable	The HKEY_LOCAL_MACHINE key was found to be writable by the "Everyone" group. This key should never be writable by the "Everyone" group under any circumstances, and indicates that the system may have been tampered with.
18014	Registry HKEY_CLASSES_ROOT Key writable	The HKEY_CLASSES_ROOT key was found to be writable by the "Everyone" group. This key should never be writable by the "Everyone" group under any circumstances, and indicates that the system may have been tampered with.
18015	Password Filter Registry Key Changed	The target host was found to have a modified value for the alternate security provider registry key. This indicates the possibility that a Trojan horse has been installed on the system to gather users' passwords when they are changed. If this key can be changed by a user, it can be modified to point to another DLL which can be used to gather passwords in clear text. This is a DLL which normally exists only in a Netware environment. A false FPNWCLNT.DLL can be stored in the %systemroot%\system32 directory which collects passwords in plain text. If an alternate provider has been intentionally installed, this test can produce a false positive. Microsoft mistakenly shipped Windows NT 4.0 with the Notification Packages value set to FPNWCLNT. This value allows any user with write permissions to the %systemroot%\system32 directory to copy in a DLL file to gather passwords.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
18016	Mail Reader Mime Bug	MIME is an encoding scheme that allows flexible encoding of various file types, such as audio messages, graphics and binary files, to be sent encoded in text. Several mail reading packages which support the MIME encoding have been found to have flaws in the parsing of the MIME header fields. During parsing, the programs copy more data into local buffers than the buffers were set up to receive resulting in corruption of internal program data. This module detects which versions of Microsoft Outlook Express, Microsoft Outlook98 or Netscape are installed on the machine through the registry. If a vulnerable mail reader is found, it is reported. It is not possible to detect if the Outlook98 patch has been applied remotely through the registry. As a result, this vulnerability will always be detected if Outlook98 is installed. You should verify that the security patch has been installed on any machine that is reported to have Outlook98 installed. Additionally this module cannot distinguish between vulnerable and non-vulnerable versions of the 4.5b1 release of Netscape. It will flag these as potentially vulnerable. You should verify that the latest version of the 4.5b1 release has been installed. In order to work properly, this module must run with the privileges of the administrator. It either needs to be run as the domain administrator in the domain the scanned machine is in, or it must know the administrator account name and password. To discover the administrator account name, module 18010 (Windows NT User ID Guessing) must have run successfully, or the administrator account must be Administrator. To discover the administrator password, module 18013 (Windows NT Password Grinding through IPC\$) must have successfully guessed the administrator password.
18017	Unsafe SNMP Registry Permissions	The permissions on the registry key containing the SNMP agent's configuration were found to be unsafe. By default, all system users are able to access the SNMP configuration.
18018	Unsafe Run Registry Key Permissions	The permissions on the Run registry key were found to allow write access by Everyone. This access allows all users and guests to add an entry to the registry, which causes a program to be executed when anyone logs into the system.
18019	Unsafe RunOnce Registry Key Permissions	The permissions on the RunOnce registry key were found to allow write access by Everyone. This access allows all users and guests to add an entry to the registry, which causes a program to be executed when anyone logs into the system.
18020	Unsafe Uninstall Registry Key Permissions	The permissions on the Uninstall registry key were found to allow write access by Everyone. This access allows all users and guests to add an entry to the registry, which causes a program to be executed when a user attempts to remove an application from the system.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
18021	NDIS 4.0 bit set for "promiscuous" mode	The target host was found to have an NDIS 4.0 Driver with its LocalOnly bit set to 1. This enables the host to enter a "promiscuous-like mode" without using the real NIC's Promiscuous mode. This is indicative of a host running a sniffer application such as Microsoft's Network Monitor.
18022	Weak protection found on base objects (C2)	The target host was found to have weak protection on the system's base objects. This is a C2 level compliance check. Tightening security on resources like COM1 and printers may be of value on a computer containing particularly sensitive data.
18023	Suspicious use of Win 3.1 File System 8.3 'short' names	The target host was found to to have a suspicious choice of prohibiting long file names. For this particular version of Windows this choice may indicate the use of a low-level disk formatter or other primitive executable that relies upon strict DOS FAT's. Some anti-virus programs and disk-tools legitimately need this setting.
18024	Unable to access IPC\$ or Registry	CyberCop Scanner was unable to obtain full access to the target host's IPC\$ share, or the Windows NT registry. Many of the policy checks in the scanner require access to the IPC\$ share or to the registry of the machine being scanned. Without the proper access, some checks will not be able to detect vulnerabilities on the remote machine. This module provides a warning specifying when access to the IPC\$ share, the HKEY_LOCAL_MACHINE registry hive or the HKEY_USERS registry hive was not granted. This indicates that a complete audit of the target system may not have been performed. This can occur if the account the scan is being run from does not have access to the machine being scanned or if the account does not have sufficient permission to access the remote resources. This may also indicate that the machine is a standalone system, or is not part of the same Windows NT domain from which the scan is being performed. If access to the registry was not obtained, it may also indicate that the target system is not a Windows NT system.
18025	IP packet forwarding is enabled	The target host was found to have IP packet forwarding enabled. This indicates the possibility of this system being used as a gateway between two lans.
18026	Auditing configured for base objects	The target host was found to have Auditing configured for base objects. An Administrator may use this to audit certain system objects not commonly known by users (i.e., they are known to software engineers). Only files and directories in NTFS partitions can be audited, and it is only access that is auditable, not intent. This setting may cause suspicion as it could be an attempt to discover internal security measures. Though it is likely a misconfiguration and should be turned off for performance reasons.
18027	TCP/IP Security not enabled	The target host was found to not have the "Security" setting in the Start/Control Panel/Network/Protocols/TCPIP Protocol/Properties/Advanced/Enable Security enabled.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
18028	Hard error mode set to suppress Messages and/or Audit-logging	The target host was found to prevent the displaying and/or logging of hard errors. This may be a concern if this host is not a secured system running authorized company applications. Typically a computer set to reboot in an un-attended mode should be physically secure and very tightly controlled.
18029	Unsecure COM reference counting	The target host was found to allow AddRef/Release invocations to be unsecure for applications that do not call CoInitializeSecurity.
18030	Suspicious COM default authentication level	The target host was found to have a non-default value for the LegacyAuthenticationLevel setting. This setting determines an authentication level for COM applications that do not call CoInitializeSecurity.
18031	MDAC settings may allow Privilege Elevation attack	The target host was found to have the IIS's Server settings, in particular those configuring the Microsoft Data Access Components (MDAC), to be in an un-safe mode. You should also check the version of the installed MDAC system. 1.5 is installed by default with the NT 4.0 Option Pack. At the time of this writing MDAC was available in version 2.1. Check the version strings inside the MSDADC.dll and OleDb32.dll's according to the Security Bulletin at: http://www.microsoft.com/security/bulletins/ms99-025faq.asp
20000	SNMP/Network Management	
20001	SNMP Community check	This module attempts to talk to a hosts SNMP server using some commonly used community names. If a successful connection is made the community is probed to see if it is read-only or read-write.
20010	SNMP MIB-II Miscellaneous data	This module gathers miscellaneous information from the SNMP daemon with the community name provided in the configuration file. This module retrieves information that is available to an attacker who has read access to SNMP. This module uses the community name specified in the configuration file and does not attempt to guess the community name. A separate SNMP community module is provided to probe for SNMP access.
20011	SNMP MIB-II TCP table	This module retrieves the TCP connection table from the SNMP daemon with the community name provided in the configuration file. This module retrieves information that is available to an attacker who has read access to SNMP. This module uses the community name specified in the configuration file and does not attempt to guess the community name. A separate SNMP community module is provided to probe for SNMP access.
20012	SNMP MIB-II UDP table	This module retrieves the table of listening UDP ports from the SNMP daemon with the community name provided in the configuration file. This module retrieves information that is available to an attacker who has read access to SNMP. This module uses the community name specified in the configuration file and does not attempt to guess the community name. A separate SNMP community module is provided to probe for SNMP access.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
20013	SNMP MIB-II Interface Table	This module retrieves the table of network interfaces from the SNMP daemon with the community name provided in the configuration file. This module retrieves information that is available to an attacker who has read access to SNMP. This module uses the community name specified in the configuration file and does not attempt to guess the community name. A separate SNMP community module is provided to probe for SNMP access.
20014	SNMP MIB-II Address table	This module retrieves the table of IP addresses from the SNMP daemon with the community name provided in the configuration file. This module retrieves information that is available to an attacker who has read access to SNMP. This module uses the community name specified in the configuration file and does not attempt to guess the community name. A separate SNMP community module is provided to probe for SNMP access.
20015	SNMP MIB-II ARP table	This module retrieves the ARP table (which contains IP address to hardware address translations) from the SNMP daemon with the community name provided in the configuration file. This module retrieves information that is available to an attacker who has read access to SNMP. This module uses the community name specified in the configuration file and does not attempt to guess the community name. A separate SNMP community module is provided to probe for SNMP access.
20016	SNMP MIB-II Routing table	This module retrieves the IP routing table from the SNMP daemon with the community name provided in the configuration file. This module retrieves information that is available to an attacker who has read access to SNMP. This module uses the community name specified in the configuration file and does not attempt to guess the community name. A separate SNMP community module is provided to probe for SNMP access.
20020	SNMP LANMAN Miscellaneous information	This module retrieves miscellaneous information in the LANMAN MIB from the SNMP daemon with the community name provided in the configuration file. This module retrieves information that is available to an attacker who has read access to SNMP. This module uses the community name specified in the configuration file and does not attempt to guess the community name. A separate SNMP community module is provided to probe for SNMP access.
20022	SNMP LANMAN Service table	This module retrieves the LANMAN table of services from the SNMP daemon with the community name provided in the configuration file. This module retrieves information that is available to an attacker who has read access to SNMP. This module uses the community name specified in the configuration file and does not attempt to guess the community name. A separate SNMP community module is provided to probe for SNMP access.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
20023	SNMP LANMAN Shares	This module retrieves the table of LANMAN shares from the SNMP daemon with the community name provided in the configuration file. This module retrieves information that is available to an attacker who has read access to SNMP. This module uses the community name specified in the configuration file and does not attempt to guess the community name. A separate SNMP community module is provided to probe for SNMP access.
20024	SNMP LANMAN Users	This module retrieves the table of LANMAN users from the SNMP daemon with the community name provided in the configuration file. This module retrieves information that is available to an attacker who has read access to SNMP. This module uses the community name specified in the configuration file and does not attempt to guess the community name. A separate SNMP community module is provided to probe for SNMP access.
20030	SNMP SunMib Process Table	This module retrieves the process table from the SNMP daemon with the community name provided in the configuration file. This module retrieves information that is available to an attacker who has read access to SNMP. This module uses the community name specified in the configuration file and does not attempt to guess the community name. A separate SNMP community module is provided to probe for SNMP access.
21000	Network Port Scanning	
21001	TCP port scanning	This check scans a target host for listening TCP ports.



Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
21002	UDP scanning check	<p>This check scans a target host for listening UDP ports. Scanning for active UDP ports is very difficult to perform reliably. This is due to the fact that UDP is a connectionless protocol, and there is no reliable indication whether or not a connection has been established. There are 2 primary methods used to scan for listening UDP ports: 1. Sending data to a UDP port, and awaiting a response from that port. 2. Sending data to a UDP port, and awaiting an ICMP port unreachable message, indicating that this port is NOT active. This allows us to build a listing of ports which may be active (if no port unreachable message is received from that port). There are problems when using both methods. When using method 1 and sending random data to each UDP port, many services will not respond if they cannot recognize the data. This results in being unable to detect many UDP servers which may be running. Using method 2 is reliable if we can ensure that two conditions are met: 1. No ICMP port unreachable messages are lost in transit. 2. The host reliably returns an ICMP port unreachable packet for every port that is inactive. This varies from operating system to operating system, in that certain operating systems implement thresholds to prevent themselves from sending out too many ICMP port unreachable messages in a period of time. Examples of this threshold have been found in versions of Linux and Solaris. CyberCop Scanner attempts to determine the best method for scanning a host for listening UDP servers. It's first choice is to scan by sending data and watching for ICMP unreachable messages. CyberCop Scanner will determine whether this is possible by first attempting this on ports 45000-45009. If CyberCop Scanner receives back all 10 ICMP port unreachable messages, it will use this method to scan for active UDP services, and assumes that the host reliably returns ICMP port unreachable messages. If this test fails, then method 1 is used, and data is sent to each port, awaiting a response. If method 2 was used, CyberCop Scanner will attempt to verify results by sending 2 more sets of data packets, and ensuring that the host is not returning ICMP port unreachable messages for ports which were found to be active earlier. This is an attempt to ensure that if any ICMP port unreachable packets were lost in transit, we do not falsely report listening ports. The results from this scan are fairly reliable when scanning on the local network, however will vary on long haul networks. Filtering routers will also cause results to vary. Note that this module can cause inferior routing software to fail. This module safely evaluates all major network operating systems.</p>

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
21003	TCP SYN port scanning	This check can be used as a much faster alternative to regular TCP port scanning. This check scans a target host for listening TCP ports in much the same way as the regular TCP port scanning, however does so by sending a packet to initiate a connection and watching for a response. The difference in using this method is that a complete connection to the remote host is not actually opened. The drawback in using this method is that it may be unreliable due to packet loss on the network.
21004	TCP ACK port scanning	This check can be used as a much faster alternative to regular TCP port scanning. This check scans a target host for listening TCP ports by observing how the target replies to a TCP ACK packet. Because the target host replies differently when an ACK is sent to a listening port than when an ACK is sent to a non-listening port, the scanner can infer which ports are being listened on. Because ports are checked without actually initiating a TCP connection, this type of scan is sometimes referred to as a "stealth" scan. The drawback in using this method is that it may be unreliable due to packet loss on the network and differing behavior of different target systems. This check may not work at all against newer versions of many operating systems.
21005	TCP FIN port scanning	This check can be used as a much faster alternative to regular TCP port scanning. This check scans a target host for listening TCP ports by observing how the target replies to a TCP FIN packet. Because the target host replies only when a FIN is sent to a non-listening port, and not when a FIN is sent to a listening port, the scanner can infer which ports are being listened on. Because ports are checked without actually initiating a TCP connection, this type of scan is sometimes referred to as a "stealth" scan. The drawback in using this method is that it may be unreliable due to packet loss on the network and differing behavior of different target systems. Because this method assumes that a target port is listening whenever a reply is not received, it is particularly prone to packet loss. As a result this scan may mistakenly report some non-listening ports as being active.
21006	RPC Scanning Direct	The RPC scanning direct check performs a UDP RPC scan of the remote host, attempting to find services by bypassing the portmapper or rpcbind. In many instances, the portmapper (port 111), which translates RPC program numbers to port numbers, is being filtered at an organization's filtering device or firewall. By directly scanning for RPC services, it is still possible to obtain a full listing of RPC services running on the remote host, and then contact them directly rather than querying the portmapper first. This check is unreliable over long haul networks, due to the unreliability of the UDP transport layer. In the case where this check is being run over a long haul network, some RPC programs which are actually running, may not appear in the scan results.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
21007	FTP bounce port scan	This module determines which TCP ports are alive on the remote host by utilizing the remote FTP server to attempt to connect to TCP ports. This module utilizes the FTP bounce attack to determine which TCP ports are active on the remote network.
22000	Windows NT - Browser Zone Policy	
22001	Internet Explorer Zone - Download unsigned ActiveX	The user's ActiveX security setting was found to be set less securely than the security policy indicates. This ActiveX setting defines whether or not unsigned ActiveX applications should be downloaded and executed.
22002	Internet Explorer Zone - Script safe ActiveX	The user's ActiveX security setting was found to be set less securely than the security policy indicates. This ActiveX setting defines whether or not safe ActiveX controls should be scripted.
22003	Internet Explorer Zone - Script unsafe ActiveX	The user's ActiveX security setting was found to be set less securely than the security policy indicates. This ActiveX setting defines whether or not unsafe ActiveX controls should be scripted.
22004	Internet Explorer Zone - Download signed ActiveX	The user's ActiveX security setting was found to be set less securely than the security policy indicates. This ActiveX setting defines whether or not signed ActiveX controls should be downloaded.
22005	Internet Explorer Zone - Run ActiveX	The user's ActiveX security setting was found to be set less securely than the security policy indicates. This ActiveX setting defines whether or not safe ActiveX controls should be run.
22006	Internet Explorer Zone - Authentication methods	The user's authentication setting was found to be set less securely than the security policy indicates. This setting specifies which authentication techniques are used over the network when accessing a remote server.
22007	Internet Explorer Zone - Font downloads	The user's Font download security settings were found to be set less securely than the security policy indicates. This option defines whether or not new fonts should be downloaded.
22008	Internet Explorer Zone - File downloads	The user's file download security settings were found to be set less securely than the security policy specifies. The file download settings specify whether or not files can be downloaded from the specified zone and stored on the user's system.
22009	Internet Explorer Zone - Java permissions	The user's Java permissions were found to be set less securely than the security policy specifies. Java security can be configured in 5 ways: - Medium safety - Low safety - High safety - Disabled - Custom
22010	Internet Explorer Zone - Software channel permissions	The user's Software channel permissions were found to be set less securely than the security policy specifies. Java security can be configured in 5 ways: - Medium safety - Low safety - High safety
22011	Internet Explorer Zone - IFRAME application launching	The user's IFRAME application launching security settings were found to be set less securely than the security policy specifies. This setting defines whether or not applications can be launched from an IFRAME (Inline Frame).

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
22012	Internet Explorer Zone - Desktop item installation	The user's desktop item installation setting is set less securely than the security policy specifies. This setting defines whether desktop items can be installed via an HTML page.
22013	Internet Explorer Zone - Submit non-encrypted form data	The 'Submit non-encrypted form data' setting defines whether or not form data can be submitted via a non-encrypted connection to a web server. This module determines whether the user's security configuration is less secure than the defined security policy.
22014	Internet Explorer Zone - Drag and drop	This user's drag and drop security settings were found to be set less securely than the security policy specifies. This security setting specifies whether items can be drag and dropped in the specified zone.
22015	Internet Explorer Zone - Java scripting	This user's java scripting security setting was found to be set less securely than the security policy specifies. This setting defines whether Java scripting is supported and whether or not to execute Java scripts.
22016	Internet Explorer Zone - Active scripting	The 'Active scripting' security setting defines whether or not Active scripting is supported in the specified zone. This module determines whether the user's security configuration is less secure than the defined security policy.
22017	Internet Explorer - Invalid site certificates option warning	If enabled, this module will check for any users who have turned the Internet Explorer 'warn about invalid site certificates' option off. This option warns users that they are connecting to an SSL site that does not have a valid site certificate, which may indicate that the page being viewed isn't the legitimate page the user requested.
22018	Internet Explorer - Changing between secure/insecure page warning	The specified user was found to have the Internet Explorer "Warn if changing between secure and not secure mode" option turned off. This option warns users when they are connected to a secure (SSL) page and are following a link to a non-secure page.
22019	Internet Explorer - Cookie security settings	The specified user was found to have the 'allow cookies' option set to a different value than specified in the security policy configuration. This option may be set to disallow the use of cookies entirely, to always allow the use of cookies, or to allow cookies but present a warning when they are used.
22020	Internet Explorer - Form submission redirection warning	The specified user was found to have the Internet Explorer 'warn if forms submit is being redirected' option was found to be off. This option warns the user when they submit a form and the page to which they submitted the form presents a redirect to another page.
22021	Internet Explorer - Do not save encrypted pages to disk option	The specified user was found to have the Internet Explorer "Do not save encrypted pages to disk" option turned off. This option prevents Internet Explorer from caching secure (SSL) pages on the local disk.
22022	Internet Explorer - Java logging disabled	The target user's Java Logging was found to be disabled. By having this feature disabled, Java events are not logged and therefore no record of Java activity is kept.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
23000	Windows NT - Privilege Enumeration	
23001	Privilege - Act as part of the operating system.	A user or group has been identified to possess Act as part of the operating system privileges. This privilege is normally not granted to any user or group. This privilege allows a process to perform as a secure, trusted part of the operating system. Only some subsystems are granted this right.
23002	Privilege - Add workstations to the domain	A user or group has been identified to possess the privilege to add a workstation to a particular domain. This right is meaningful only on domain controllers. This privilege is normally not granted to any user or group.
23003	Privilege - Back up files and directories	A user or group has been identified to possess the privilege to backup files and directories. This right bypasses any file and directory permissions and allows the user full access to all files. This privilege is normally only allowed to members of the Administrators, Backup Operators, and Server Operators groups.
23004	Privilege - Bypass traverse checking.	A user or group has been identified to possess the privilege to bypass traverse checking. This privilege is given to all users and allows users to change into directories and access files and subdirectories even if the user has no permission to access parent directories.
23005	Privilege - Change system time privilege	A user or group has been identified to possess Change system time privileges. This privilege allows a user to set the internal clock of the computer. This privilege is normally only allowed to members of the Administrators, Power Users, and Server Operators groups.
23006	Privilege - Create Pagefile Privilege	A user or group has been identified to possess the privilege to create system page files. This privilege allows users to create new page files where system virtual memory will be stored. This privilege is normally only allowed to members of the Administrators group.
23007	Privilege - Create a token object	A user or group has been identified to possess the privilege to create access tokens. This privilege is only allowed by the Local Security Authority (LSA). This privilege is normally not granted to any user or group.
23008	Privilege - Create Permanent Shared Objects	A user or group has been identified to possess the privilege to create permanent shared objects. This privilege allows users to create special shared objects that are used within Windows NT. An example of this is the \Device object. This privilege is normally not granted to any user or group.
23009	Privilege - Debug Programs	A user or group has been identified to possess privilege to debug programs. This privilege allows the debugging of low level system objects such as program threads. This privilege is normally only allowed to members of the Administrators group.
23010	Privilege - Force shutdown from a remote system	A user or group has been identified to possess the privilege to shut the system down from a remote system. This privilege allows the user to shutdown the system at will. This privilege is normally only allowed to members of the Administrators, Power Users, and Server Operators groups.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
23011	Privilege - Generate Security Audits	A user or group has been identified to possess the privilege to generate security audits. This privilege is normally used by low level system processes to generate security audit messages, which are stored in the system security log. This privilege is normally not granted to any user or group.
23012	Privilege - Increase Quota Privilege	A user or group has been identified to possess the privilege to increase quotas. This privilege is not used in the current implementation of Windows NT, however may be implemented in future revisions.
23013	Privilege - Increase Scheduling Priority	A user or group has been identified to possess the privilege to increase the priority of a process. This privilege is normally only allowed to members of the Administrators and Power Users groups.
23014	Privilege - Load and unload device drivers	A user or group has been identified to possess the privilege to load and unload device drivers. This privilege allows a user to install and remove device drivers from the system. This privilege can allow a user to gain Administrator access. This privilege is normally only allowed to members of the Administrators group.
23015	Privilege - Lock pages in memory	A user or group has been identified to possess the privilege to lock pages in memory. This privilege allows a user to lock pages in memory so that they cannot be paged out by the virtual memory system. This prevents the pages from ever being removed from memory to be stored in the system page file. This privilege is normally not granted to any user or group.
23016	Privilege - Manager auditing and security log	A user or group has been identified to possess the privilege to manage the auditing system and the security logs. This allows the user to specify what type of resource access is to be audited (such as file access), and to view and clear the security log. This does not, however, allow the user to change auditing events via the User Manager -> Audit menu. This privilege is normally only allowed to members of the Administrators group.
23017	Privilege - Modify firmware environment variables	A user or group has been identified to possess the privilege to modify system environment variables stored in non-volatile RAM. The system must support this type of configuration for this privilege to be significant. This privilege is normally only allowed to members of the Administrators group.
23018	Privilege - Profile Single Process	A user or group has been identified to possess the privilege to perform profiling (performance sampling) on a process. This privilege is normally only allowed to members of the Administrators and Power Users groups.
23019	Privilege - Profile System Performance	A user or group has been identified to possess the privilege to perform profiling on the entire system (performance monitoring). This privilege is normally only allowed to members of the Administrators group.
23020	Privilege - Replace a process-level token	A user or group has been identified to possess the privilege to replace process level tokens. This allows a user to modify a processes security access token. This privilege is normally used only by the system and is not granted to any user or group.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
23021	Privilege - Restore files and directories	A user or group has been identified to possess the privilege to restore files and directories. This permissions allows the user to restore from backup, files and directories to the system. This privilege overrides any file and directory access level restrictions. This privilege is normally only allowed to members of the Administrators, Backup Operators, and Server Operators groups.
23022	Privilege - Take ownership of files or other objects	A user or group has been identified to possess the privilege to take ownership of files or objects. This right bypasses any permissions that are in place to protect the object, and give ownership to the specified user. This privilege is normally only allowed to members of the Administrators group.
23023	Backup Operators Group - Check for users that do not belong by default	A user or users have been identified as not belonging to the Backup Operators Group by default.
23024	Power Users Group - Check for users that do not belong by default	A user or users have been identified as not belonging to the Power Users Group by default.
23025	Print Operator Group - Check for users that do not belong by default	A user or users have been identified as not belonging to the Print Operator Group by default.
23026	Replicator Group - Check for users that do not belong by default	A user or users have been identified as not belonging to the Replicator Group by default.
23027	System Operator Group - Check for users that do not belong by default	A user or users have been identified as not belonging to the System Operator Group by default.
23028	Account Operators Group - Check for users that do not belong by default	A user or users have been identified as not belonging to the Account Operators Group by default.
23029	Administrators Group - Check for users that do not belong by default	A user or users have been identified as not belonging to the Administrators Group by default.
23030	Guests Group - Check for users that do not belong by default	A user or users have been identified as not belonging to the Guests Group by default.
23031	Domain Administrators Group - Check for users that do not belong by default	A user or users have been identified as not belonging to the Domain Administrators Group by default.
24000	Windows NT - Local System Policy	
24001	Legal Notice - No Legal Caption Specified	The security policy indicates that a legal caption must be displayed for users when a logon is initiated. This host does not have a legal caption present. Windows NT has the ability to display a caption containing text of your choice, notifying potential users that they can be held legally liable if they attempt to use the computer without valid authorization. The absence of such a message may be construed as an invitation to enter the computer system without authorization.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
24002	Legal Notice - Legal Caption does not match Policy	The security policy indicates that a specific network wide legal caption must be specified for all systems. This host does not contain the legal caption specified by the security policy.
24003	Legal Notice - No Legal Text Specified	The security policy indicates that legal text must be displayed for users when a logon is initiated. This host does not have legal text present. Windows NT has the ability to display legal text containing text of your choice, notifying potential users that they can be held legally liable if they attempt to use the computer without valid authorization. The absence of such a message may be construed as an invitation to enter the computer system without authorization.
24004	Legal Notice - Legal Text does not match Policy	The security policy indicates that specific network wide legal text must be specified for all systems. This host does not contain the legal text specified by the security policy.
24005	Event Log - Application Event Log Not Restricted	This host does not restrict access to the Application Event Log by Guest and Null-user logons. This situation allows arbitrary network users to access this log information.
24006	Event Log - Security Event Log Not Restricted	This host does not restrict access to the Security Event Log by Guest and Null-user logons. This situation allows arbitrary network users to access this log information. This is an unusual situation as, unlike the Application and System Event logs, the Security log is protected by the default installation.
24007	Event Log - System Event Log Not Restricted	This host does not restrict access to the System Event Log by Guest and Null-user logons. This situation allows arbitrary network users to access this log information.
24008	Restrict Print Driver - Secure Print Driver Installation	The addition of printer drivers should be restricted to Administrators and Print Operators (on server), or Power Users (on workstation). This host does not currently enforce this restriction.
24009	Restrict Schedule Service - Secure Schedule Service (AT command)	This host was found to allow System Operators to submit AT commands. Scheduled commands are run in the context of the Schedule Service itself, the System context, which provides even more privilege than Administrator access. By utilizing the schedule service, it is possible for authorized users to obtain increased privileges to the system.
24010	Restrict Last User - Displaying of Last Logged in User	The name of the last user who utilized the system should not be displayed in the Logon box. This is done by default to make it more convenient to logon to the system. This host currently displays the name of the last logged in user in the Logon box. This is a concern if the Administrator account has been renamed, and is frequently used. Users walking by can obtain the new Administrator name by looking at the Logon window.
24011	Restrict Shutdown - Prevent System Shutdown from Logon Window	System Shutdown should not be allowed from the initial system Logon Window. By allowing the Shutdown process from the Logon Window, any user walking by is able to shut the system down, without logging in.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
24012	Restrict Floppy - Prevent Process Access to the Floppy Disk Drive	Floppy Disk Drive access should be allowed only for the currently logged on interactive user. The host currently allows any process to access the Floppy Disk Drive, thus allowing any process, even those not owned by the current user, to access the Floppy Disk Drive.
24013	Restrict CDROM - Prevent Process Access to the CDROM Drive	CDROM Drive access should be allowed only for the currently logged on interactive user. The host currently allows any process to access the CDROM Drive, thus allowing any process, even those not owned by the current user, to access the CDROM Drive.
24014	Clear System Page File during System Shutdown	The system page file should be cleared during a clean system shutdown. The host does not currently enforce this policy. The system page file is used by the Windows NT virtual memory manager to swap pages of processes from memory to disk when they are not being used.
24015	Disable Caching of Logon Credentials	The caching of logon credentials should be disabled during interactive logon. The host does not currently enforce this policy. Windows NT by default caches the last logon credentials for a user who has logged on interactively to the system. This allows the system to function and allow logons if the system were to be disconnected from the network, or the Primary Domain Controller were to become unavailable.
24016	Subsystems - POSIX Subsystem Enabled	The POSIX subsystem should be disabled. The host does not currently disable the POSIX subsystem.
24017	Subsystems - OS/2 Subsystem Enabled	The OS/2 subsystem should be disabled. The host does not currently disable the OS/2 subsystem.
24018	Registry - Registry Association with REGEDIT.EXE	Registry files are currently associated with the registry editor.
24019	Screen Saver Lockout Not Enabled	The screen saver lockout functionality should be enabled. The target host does not currently enforce this. The screen saver lockout forces the user to enter their logon password once the screen saver has been activated.
24020	Restrict Autorun - Prevent Automatic Execution of CDROM	The Autorun should be disabled on the CDROM Drive. The host currently has Autorun enabled.
24022	Log Policy - Application Log Maximum Size	The maximum size of the Application Log on the target host does not match the policy setting. The maximum size specifies how large the application log can grow before entries are over-written, or the log is declared as full.
24023	Log Policy - Application Log Retention Period	The retention period of the Application Log on the target host does not match the policy setting. The retention period specifies how long log entries are to be kept before being over-written.
24024	Log Policy - Security Log Maximum Size	The maximum size of the Security Log on the target host does not match the policy setting. The maximum size specifies how large the security log can grow before entries are over-written, or the log is declared as full.
24025	Log Policy - Security Log Retention Period	The retention period of the Security Log on the target host does not match the policy setting. The retention period specifies how long log entries are to be kept before being over-written.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
24026	Log Policy - System Log Maximum Size	The maximum size of the System Log on the target host does not match the policy setting. The maximum size specifies how large the system log can grow before entries are over-written, or the log is declared as full.
24027	Log Policy - System Log Retention Period	The retention period of the System Log on the target host does not match the policy setting. The retention period specifies how long log entries are to be kept before being over-written.
25000	Windows NT - Auditing and Password Policy	
25001	Auditing - Restart, Shutdown, and System Events - Success	The auditing of successful Restart, Shutdown and System events was found to be disabled on the target host. Your security policy defines that these events should be audited. Auditing of Restart, Shutdown, and System events allows recording of systems starts, shutdowns, and restarts.
25002	Auditing - Restart, Shutdown, and System Events - Failure	The auditing of failed Restart, Shutdown and System events was found to be disabled on the target host. Your security policy defines that these events should be audited. Auditing of Restart, Shutdown, and System events allows recording of systems starts, shutdowns, and restarts.
25003	Auditing - Logon and Logoff Events - Success	The auditing of successful Logon and Logoff events was found to be disabled on the target host. Your security policy defines that these events should be audited. Auditing of successful Logon and Logoff Events allows tracking of both local and remote user logons, as well as logons to use the system's resources. Auditing of successful Logon and Logoff events allows tracking of system usage, as well as identifying the misuse of accounts.
25004	Auditing - Logon and Logoff Events - Failure	The auditing of failed Logon and Logoff events was found to be disabled on the target host. Your security policy defines that these events should be audited. Auditing of failed Logon and Logoff Events allows the administrator to identify brute-force password attacks, where an attacker attempts to guess a username and password via repeated logon requests.
25005	Auditing - File and Object Access Events - Success	The auditing of successful File and Object Access events was found to be disabled on the target host. Your security policy defines that these events should be audited. Auditing of File and Object Access Events can be utilized to track down users accessing sensitive files on the target host.
25006	Auditing - File and Object Access Events - Failure	The auditing of failed File and Object Access events was found to be disabled on the target host. Your security policy defines that these events should be audited. Auditing of File and Object Access Events can be utilized to track down users accessing sensitive files on the target host.
25007	Auditing - Use of User Rights - Success	The auditing of successful Use of User Rights was found to be disabled on the target host. Your security policy defines that these events should be audited. By auditing the Use of User Rights, the Administrator can track the misuse of privileges by authorized users.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
25008	Auditing - Use of User Rights - Failure	The auditing of failed Use of User Rights was found to be disabled on the target host. Your security policy defines that these events should be audited. By auditing the Use of User Rights, the Administrator can track the misuse of privileges by authorized users.
25009	Auditing - Process Tracking - Success	The auditing of successful Processes was found to be disabled on the target host. Your security policy defines that these events should be audited. By auditing the Processes on the host, you can track program activation, handle duplication, indirect object access, and process exit. This functionality allows an Administrator to identify unusual processes running on their systems.
25010	Auditing - Process Tracking - Failure	The auditing of failed Processes was found to be disabled on the target host. Your security policy defines that these events should be audited. By auditing the Processes on the host, you can track program activation, handle duplication, indirect object access, and process exit. This functionality allows an Administrator to identify unusual processes running on their systems.
25011	Auditing - Security Policy Changes - Success	The auditing of successful Security Policy Changes was found to be disabled on the target host. Your security policy defines that these events should be audited. Auditing Security Policy Changes allows an administrator to keep track of any changes made to the user rights configuration, or the audit policy configuration on the target host.
25012	Auditing - Security Policy Changes - Failure	The auditing of failed Security Policy Changes was found to be disabled on the target host. Your security policy defines that these events should be audited. Auditing Security Policy Changes allows an administrator to keep track of any changes made to the user rights configuration, or the audit policy configuration on the target host.
25013	Auditing - User and Group Management Events - Success	The auditing of successful User and Group Management Events was found to be disabled on the target host. Your security policy defines that these events should be audited. Auditing of User and Group Management Events allows tracking of any user account or group creations, changes, or deletions, any user accounts that are renamed, disabled, or enabled, as well as all password changes.
25014	Auditing - User and Group Management Events - Failure	The auditing of failed User and Group Management Events was found to be disabled on the target host. Your security policy defines that these events should be audited. Auditing of User and Group Management Events allows tracking of any user account or group creations, changes, or deletions, any user accounts that are renamed, disabled, or enabled, as well as all password changes.
25015	Auditing - Shut Down When Audit Log Full	The security policy indicates that hosts should shut down when their audit log becomes full. This host has not been configured to do so. If this option is not chosen, important security events may not be logged. If this option is chosen, when the audit log is full, the system reboots and causes a Blue Screen. Once rebooted, only the Administrator is allowed to log onto the machine (locally or remotely). The Administrator is then required to clean the audit log.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
25016	Account Lockout Policy - Lockout Threshold	This host was found to have an account lockout threshold value which differs from that which is defined in the security policy. The account lockout threshold defines how many invalid logon attempts can be made before the account is locked for a period of time.
25017	Account Lockout Policy - Lockout Period	This host was found to have an account lockout period value which differs from that which is defined in the security policy. The account lockout period defines how long an account will be locked out and disabled after the defined number of invalid logons.
25018	Account Lockout Policy - Lockout Window	This host was found to have an account lockout window value which differs from that which is defined in the security policy. The account lockout window defines how long the system will wait before resetting the count of the number of invalid logons back to 0. For example, if the account lockout threshold is set to 5, and there were 4 invalid logons, if the account lockout window is set to 30 minutes, and there are no other invalid logons after 30 minutes, the number of invalid logons is set to 0.
25019	Account Password Policy - Minimum Password Length	This host was found to have a minimum password length which is less than the minimum password length defined in the security policy. A short password is easier for an attacker to crack, weakening the overall security of the system.
25020	Account Password Policy - Password History	This host was found to have a password history length which is less than the minimum password history length defined in the security policy. Not enforcing, or defining a low password history length allows users to utilize passwords which they have utilized in the past. By doing this, users may open the system up to an attacker, if a previous password has been obtained.
25021	Account Password Policy - Maximum Password Age	This host was found to have a maximum password age which is greater than the maximum password age defined in the security policy. The maximum password age defines the amount of time which can pass before a user is forced to change their password to a new password. By allowing a large maximum password age, users will be forced to change their passwords less frequently, decreasing the overall security of the system.
25022	Account Password Policy - Minimum Password Age	This host was found to have a minimum password age which is less than the minimum password age defined in the security policy. The minimum password age defines the amount of time which must pass before a user can change their password again. The minimum password age mechanism is used to prevent users from circumventing the password history mechanism by changing their password repeatedly until the history mechanism has forgotten their original password. After this has occurred, the user could enter their original password again.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
25023	Account Policy - Forcibly disconnect expired users	On Primary Domain Controllers (PDC's), the 'Forcibly disconnect expired users' setting determines whether or not users are forced to disconnect from any servers on a domain when their logon hours are exceeded. If this setting is not enabled, users cannot make additional connections to the domain outside of their scheduled logon hours, but existing connections will not be terminated. This module checks to see if the 'Forcibly disconnect expired users' setting on the PDC is in violation of the configured security policy.
26000	Windows NT - Information Gathering	
26001	User Enumeration via Anonymous Logon	A listing of user accounts present on the target host was retrieved. Windows NT provides enumeration functions for enumerating users on the network. By default, Windows NT 4.0 and 3.51 allow anonymous logon users (also known as NULL session connections) to list account names.
26002	Active Users Enumeration via Anonymous Logon	A listing of logged in users on the target host was retrieved. Windows NT provides enumeration functions for enumerating users on the network. By default, Windows NT 4.0 and 3.51 allow anonymous logon users (also known as NULL session connections) to list account names.
26003	Group Enumeration via Anonymous Logon	A listing of groups present on the target host was retrieved. Windows NT provides enumeration functions for enumerating groups on the network. By default, Windows NT 4.0 and 3.51 allow anonymous logon users (also known as NULL session connections) to list group names.
26004	Share Enumeration via Anonymous Logon	A listing of shares present on the target host was retrieved. Windows NT provides enumeration functions for enumerating shares on the network. By default, Windows NT 4.0 and 3.51 allow anonymous logon users (also known as NULL session connections) to list shares.
26005	Enumerate Network Transports via Anonymous Logon	CyberCop Scanner was able to retrieve a listing of network transports which are present on the target host. Windows NT provides functions for enumerating the transports on a network. This module uses these functions to enumerate all the network transports on a machine. This provides a list of the networking transports installed on a machine as well as the hardware addresses of the network cards bound to the transports.
26006	Enumerate Active Sessions via Anonymous Logon	CyberCop Scanner was able to retrieve a listing of sessions which are active on the target host. A listing of active sessions displays all resources which are currently being accessed on the target host.
26007	User ID Guessing	Windows NT uses numeric IDs to identify users. It provides functions to resolve these identifiers into user names. These functions can be invoked remotely. This module tries to resolve a range of user ID's that administrator and user accounts are commonly assigned from. Because the administrator account retains the same ID even after being renamed, it is possible to determine the administrator account name even if it has been renamed.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
26008	Machine Info from the Registry through IPC\$ Share	NT stores most configuration information in the registry. The registry may be accessed remotely through the IPC\$ share. This module retrieves general information about an NT machine from the registry.
26009	IP Address Info from the Registry through IPC\$ Share	NT stores most configuration information in the registry. The registry may be accessed remotely through the IPC\$ share. This module retrieves information about the network interfaces in a machine and the addresses assigned to them.
26010	Enumerate RPC Bindings (EPDUMP)	This check will gather information about a remote machine by walking through the table of all bound RPC endpoints and listing them. This provides some information about what RPC services are running on the machine and which are accessible remotely through IP or over SMB.
27000	Intrusion Detection System Verification	
27001	IDS Single Out-of-Order TCP Segment Test	This test determines whether a network intrusion detection system is capable of reconstructing data from network transactions when the packets comprising those transactions are sent out-of-order. Real TCP/IP network software is capable of handling arbitrarily ordered packets; network intrusion detection software is frequently unable to do so.
27002	IDS Baseline (Single-Segment)	This test determines whether a network intrusion detection system is appropriately configured to detect attacks in TCP network traffic.
27003	IDS TCB Desynchronization Test (RST)	This test attempts to "desynchronize" an intrusion detection system from a TCP connection being used to carry out an attack. By creating a false TCP connection prior to carrying out a real attack, this test attempts to convince an IDS that the attack-bearing connection is entirely invalid, thus preventing it from monitoring the data exchanged in the connection. This specific test functions by opening a connection, immediately resetting it, and opening a new connection in its place. A real TCP/IP stack will appropriately handle the new connection; broken IDS software that does not correctly deal with TCP connection resets will not detect the new connection.
27004	IDS All Out-of-Order TCP Segment Test	This test determines whether a network intrusion detection system is capable of reconstructing data from network transactions when the packets comprising those transactions are sent out-of-order. Real TCP/IP network software is capable of handling arbitrarily ordered packets; network intrusion detection software is frequently unable to do so.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
27005	IDS TCP Sequence Number Verification Test (Jump-Up)	This test attempts to determine whether a network intrusion detection system adequately verifies the sequence numbers on TCP segments. Real TCP/IP network software discards TCP segments that do not bear appropriate sequence numbers. Network intrusion detection software frequently does not, and can be forced to accept bad network packets which confuse TCP analysis and allow attacks to be slipped past the system. This specific test functions by artificially increasing the sequence numbers in mid-connection. A real TCP/IP stack will discard the connection at this point; poorly functioning IDS software will not.
27006	IDS TCP Sequence Number Verification Test (Interleave)	This test attempts to determine whether a network intrusion detection system adequately verifies the sequence numbers on TCP segments. Real TCP/IP network software discards TCP segments that do not bear appropriate sequence numbers. Network intrusion detection software frequently does not, and can be forced to accept bad network packets which confuse TCP analysis and allow attacks to be slipped past the system. This specific test functions by artificially inserting a badly-sequenced duplicate TCP segment after each legitimate segment. Real TCP/IP stacks will discard the bad segments and reassemble the attack the connection contains. Poorly functioning IDS software will not.
27007	IDS IP Checksum Verification	This test attempts to determine whether an intrusion detection system correctly verifies the IP checksum carried on all IP packets. Real TCP/IP software ensures that the checksum on each packet is valid before processing it. Many network intrusion detection systems do not verify the checksum, and can thus be fooled into accepting bad packets, which confuses network traffic analysis and allows attacks to be slipped past the system.
27008	IDS TCP Checksum Verification	This test attempts to determine whether an intrusion detection system correctly verifies the TCP checksum carried on all TCP packets. Real TCP/IP software ensures that the checksum on each packet is valid before processing it. Many network intrusion detection systems do not verify the checksum, and can thus be fooled into accepting bad packets, which confuses network traffic analysis and allows attacks to be slipped past the system.
27009	IDS TCB Desynchronization Test (Data)	This test attempts to "desynchronize" an intrusion detection system from a TCP connection being used to carry out an attack. By creating a false TCP connection prior to carrying out a real attack, this test attempts to convince an IDS that the attack-bearing connection is entirely invalid, thus preventing it from monitoring the data exchanged in the connection.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
27010	IDS TCP Data-in-SYN Test	This test attempts to determine whether a network intrusion detection system correctly deals with data contained in TCP handshake packets. Real TCP/IP software, in accordance with the RFC standard for the TCP protocol, accepts data contained in SYN handshake packets. Many network intrusion detection systems do not, and data contained in SYN packets is thus invisible to these systems.
27011	IDS IP Fragment Replay	"Fragmentation" is the process by which large IP packets are broken into smaller packets for transmission over network media with packet size limitations. All real TCP/IP stacks handle fragmentation, which requires the network stack to reassemble complete IP packets from streams of fragmented packets. This test attempts to verify that a network intrusion detection system correctly reassembles complete IP packets out of IP fragment streams. This specific test attempts to confuse an intrusion detection system by "replaying" a single fragment in a stream of fragments. Real TCP/IP stacks will discard the duplicated fragment. Broken IDS software may incorrectly reassemble the entire fragment stream.
27012	IDS IP Fragmentation Test (8-Byte Tiny Fragments)	"Fragmentation" is the process by which large IP packets are broken into smaller packets for transmission over network media with packet size limitations. All real TCP/IP stacks handle fragmentation, which requires the network stack to reassemble complete IP packets from streams of fragmented packets. This test attempts to verify that a network intrusion detection system correctly reassembles complete IP packets out of IP fragment streams.
27013	IDS IP Fragmentation Test (24-byte Packets)	"Fragmentation" is the process by which large IP packets are broken into smaller packets for transmission over network media with packet size limitations. All real TCP/IP stacks handle fragmentation, which requires the network stack to reassemble complete IP packets from streams of fragmented packets. This test attempts to verify that a network intrusion detection system correctly reassembles complete IP packets out of IP fragment streams.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
27014	IDS IP Fragment Out-of-Order Test	"Fragmentation" is the process by which large IP packets are broken into smaller packets for transmission over network media with packet size limitations. All real TCP/IP stacks handle fragmentation, which requires the network stack to reassemble complete IP packets from streams of fragmented packets. This test attempts to verify that a network intrusion detection system correctly reassembles complete IP packets out of IP fragment streams. This specific test attempts to confuse an intrusion detection system by sending a single fragment out-of-order, with the marked "final" fragment sent before the last data fragment. Real TCP/IP stacks will correctly reassemble fragments regardless of the order in which they arrive. Broken network IDS software may incorrectly reassemble the entire fragment stream, especially when the final fragment appears out of order (some systems may mistakenly assume a fragment stream has been completely transmitted as soon as the final fragment appears in the stream).
27015	IDS IP Fragmentation Overlap Test	"Fragmentation" is the process by which large IP packets are broken into smaller packets for transmission over network media with packet size limitations. All real TCP/IP stacks handle fragmentation, which requires the network stack to reassemble complete IP packets from streams of fragmented packets. This test attempts to verify that a network intrusion detection system correctly reassembles complete IP packets out of IP fragment streams. This specific test attempts to confuse an intrusion detection system by sending multiple fragments of varying sizes which overlap each other. Different operating systems handle this condition in different ways. An intrusion detection system that cannot duplicate exactly the manner in which the target of an attack resolves overlapping fragments can be forced to incorrectly reassemble a fragment stream.
27016	IDS TCP Three-Way-Handshake Test	TCP connections are initiated by means of a handshake protocol, during which both sides of the connection agree to the parameters used by the connection. All TCP/IP stacks communicate over TCP only after establishing a connection with a handshake. Some network intrusion detection systems ignore the handshake entirely, and assume that any data sent over the network in a TCP packet is part of a legitimate connection. This test attempts to verify whether a network intrusion detection system actually waits for a handshake before recording data from a connection.
27017	IDS TCP ACK Flag Verification	Normally, all data exchanged in a TCP connection is sent in a TCP packet with the ACK ("acknowledge") flag set. Many TCP/IP stacks will refuse to accept data in a packet that does not bear an ACK flag. Network intrusion detection systems frequently do not verify the presence of the ACK flag, and can thus be confused into accepting data that is not actually being exchanged in an actual connection.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
27018	IDS IP Fragmentation Test (Out-of-Order Fragments)	"Fragmentation" is the process by which large IP packets are broken into smaller packets for transmission over network media with packet size limitations. All real TCP/IP stacks handle fragmentation, which requires the network stack to reassemble complete IP packets from streams of fragmented packets. This test attempts to verify that a network intrusion detection system correctly reassembles complete IP packets out of IP fragment streams. This specific test attempts to confuse an intrusion detection system by sending a single fragment out-of-order. Real TCP/IP stacks will correctly reassemble fragments regardless of the order in which they arrive. Broken network IDS software may incorrectly reassemble the entire fragment stream.
27019	IDS TCP Segment Retransmission (Inconsistent)	Individual segments in a TCP connection can be repeated. Typically, the first correctly-sequenced segment received in a connection will be accepted, and subsequent duplicate segments will be discarded. Real TCP/IP stacks handle retransmitted segments in a robust fashion by considering sequence numbers. Many intrusion detection systems fail to do so, and can be forced to accept invalid data when segments are repeated. This specific test attempts to confuse a network IDS by replaying a segment with inconsistent data. Normally the TCP/IP stack will discard the retransmitted packet, while some IDS software will accept the packet and become desynchronized.
27020	IDS TCP Segment Retransmission	Individual segments in a TCP connection can be repeated. Typically, the first correctly-sequenced segment received in a connection will be accepted, and subsequent duplicate segments will be discarded. Real TCP/IP stacks handle retransmitted segments in a robust fashion by considering sequence numbers. Many intrusion detection systems fail to do so, and can be forced to accept invalid data when segments are repeated. This specific test attempts to confuse a network IDS by replaying a single segment. A real TCP/IP stack will discard the retransmitted packet; broken IDS software will accept the packet and become desynchronized.
27021	IDS TCP Second-SYN Test	TCP connections are initiated by a handshake protocol involving TCP packets with the SYN flag set. A TCP SYN packet requests a new connection to be created, and specifies the sequence numbers for the new connection. Real TCP/IP software rejects SYN packets received after a connection has started. Broken intrusion detection system software may become confused when spurious SYN packets are received.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
27022	IDS TCP Reset Test	TCP connections are terminated by messages that request connection teardown. Real TCP/IP software closes open TCP connections when a correctly -sequenced teardown message is received; once a connection is closed, a new connection can be created using the same ports. Some broken intrusion detection systems fail to tear down connections when a teardown message is received. These systems are incapable of tracking new connections that re-use the port numbers from previously closed connections.
27023	IDS Baseline (Multiple-Segments)	This test determines whether a network intrusion detection system is appropriately configured to detect attacks in TCP network traffic.
27024	IDS TCP Sequence Number Wrapping	TCP sequence numbers are 32-bit integers. The sequence numbers of a given connection start at an effectively random number. TCP/IP network stacks are required to handle sequence number "wraparound", which occurs when the TCP sequence number exceeds the maximum number that can be expressed in 32 bits and thus wraps back to zero. Broken network intrusion detection systems fail to handle this case, and packets received after the sequence numbers wrap will be discarded.
27025	IDS TCP Overlap Test	TCP packets contain a variable amount of data. The sequence numbers on a TCP segment specify what point in the stream the data in that segment should appear at. Two TCP segments can contain conflicting data if the sequence space used by the two segments "overlap". Different TCP/IP stacks handle this rare case in different manners. A network intrusion detection system that cannot duplicate exactly the behavior of the systems it watches can be confused, and forced to see different data on the network than what is actually being exchanged.
28000	Windows NT - Service Packs (SP) and Hot Fixes (HF)	
28001	Determine if host Registry can be accessed	This check will return whether or not the Registry on this Windows host is accessible from this scanner -host.
28002	Determine the installed Service Pack revision	This check will return which service pack is installed in this Windows host. The Service Packs checked for are SP1 through (and including) SP6. If no Service Pack is installed then this check will return "No Service Pack Installed".
28005	SP1 is not installed	This check will verify that the Service Pack 1 software is installed and report a vulnerability if it is not detected.
28006	SP2 is not installed	This check will verify that the Service Pack 2 software is installed and report a vulnerability if it is not detected.
28010	SP3 (40-bit Cipher-strength) is not installed	This check will verify that the 40-bit cipher-strength (exportable) version of Service Pack 3 software is installed and report a vulnerability if it is not detected.
28011	SP3 (128-bit Cipher strength) is not installed	This check will verify that the 128-bit cipher strength (non exportable) version of Service Pack 3 software is installed and will report a vulnerability if it is not detected.
28012	SP3 is not installed	This check will verify that the Service Pack 3 software is installed and will report a vulnerability if it is not detected.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
28013	HF-SP3 Access Violation in Dns.exe Caused by Malicious Telnet Attack	This host is susceptible to an access violation (a fault) in the DNS module after malicious attack. There are also four other potential security holes that are not addressed due to the lack of this hotfix. This check will verify that this software is installed and report a vulnerability if not.
28014	HF-SP3 No Memory.dmp File Created with RAM Above 1.7 GB	This host is susceptible to failing to write a "Memory.dmp" file upon faulting. If this host has more than 1.7 gigabytes of physical memory installed this hotfix should be applied. This check will verify that this software is installed and report a vulnerability if not.
28015	HF-SP3 Performance degradation due to memory leak in ASP.DLL	This host is susceptible to performance degradation using Active Server Pages 1.0. This check will verify that this software is installed and report a vulnerability if not.
28016	HF-SP3 IBM DTTA-351010 10.1 GB Drive Capacity Is Inaccurate	The hard disk on this host may incorrectly report its available free space. This check will verify that this software is installed and report a vulnerability if not.
28017	HF-SP3 Euro Currency Not Available in Windows NT Character Sets	This host does not have a Euro Currency Symbol as part of its Western European Character Set. This check will verify that this software is installed and report a vulnerability if not.
28018	HF-SP3 GetAdmin Utility Grants Users Administrative Rights	This host is susceptible to having a malicious user run the popularly available application "Getadmin.exe" to grant normal users administrative rights by adding them to the "Administrators" group. There are also two other potential security holes that are not addressed due to the lack of this hotfix. This check will verify that this software is installed and report a vulnerability if not.
28019	HF-SP3 WinNT Lets You Paste Text into Unlock Workstation Dialog Box	This host is susceptible to a malicious user accessing the first line of clipboard-text from the locked console. This check will verify that this software is installed and report a vulnerability if not.
28020	HF-SP3 Write Cache on IDE/ATAPI Disks Is Not Flushed on Shut Down	This host is susceptible to a "blue-screen" on startup or starting-up and reporting a "dirty" volume (runs CHKDSK automatically). The blue-screen will have the following text: "STOP 0x0000007B (parameter, parameter, parameter, parameter)" "INACCESSIBLE_BOOT_DEVICE" This check will verify that this software is installed and report a vulnerability if not.
28021	HF-SP3 TCP/IP Causes Time Wait States to Exceed Four Minutes	This host is susceptible to failing to report incoming data for a short period of time while in the Winsock 2 service provider for TDI module. In TCP/IP, time wait state queue management had a problem that caused time wait states to exceed four minutes under stress. This check will verify that this software is installed and report a vulnerability if not.
28022	HF-SP3 Administrators can Display Contents of Service Account Passwords	This host is susceptible to having a malicious program display security information retained by the LSA (Local Security Authority). This includes data such as the passwords for service accounts. Additional encryption for LSA secrets is needed to properly protect passwords on this host. This check will verify that this software is installed and report a vulnerability if not.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
28023	HF-SP3 Memory Leak and STOP Screens Using Intermediate NDIS Drivers	This host is susceptible to memory leaks and STOP-screens (crashing) while executing in the NDIS driver layer. This can occur when an add-on NDIS layer (such as a filtering driver used for virus-checking or disk-compression) is installed. This check will verify that this software is installed and report a vulnerability if not.
28024	HF-SP3 Connecting to a Server is Slow over RAS Using LMHOSTS File	This host's RAS Server is susceptible to delaying users attempting to dial-in the first time by up to 90 seconds. This check will verify that this software is installed and report a vulnerability if not.
28025	HF-SP3 Xircom PC Card Fails to Function	This host is susceptible to inadvertently re-setting the "type" field on the Xircom CBE-10/100BTX Network Interface Card. This may cause the board to fail. This check will verify that this software is installed and report a vulnerability if not.
28026	HF-SP3 Invalid Operand with Locked CMPXCHG8B Instruction	The host may hang given a specific invalid (CPU) instruction. This check will verify that this software is installed and report a vulnerability if not.
28027	HF-SP3 PPTP Performance & Security Upgrade for WinNT 4.0 Release Notes	New Performance and Security upgrade features of RAS/PPTP are not applied on this host. This check will verify that this software is installed and report a vulnerability if not.
28028	HF-SP3 SecHole Lets Non-administrative Users Gain Debug Level Access	This host is susceptible to an elevation of privilege attack by a malicious program. This check will verify that this software is installed and report a vulnerability if not.
28029	HF-SP3 Group of Hotfixes for Exchange 5.5 and IIS 4.0	Several problems including a possible Access Violation during Windows NT Explorer and a security problem with IIS/ASP are not addressed on this host. This check will verify that this software is installed and report a vulnerability if not.
28030	HF-SP3 EBCDIC Characters not Properly Converted to ANSI Characters	This host is susceptible to corrupting data due to improper conversion of EBCDIC to ANSI. This check will verify that this software is installed and report a vulnerability if not.
28031	HF-SP3 Fault Tolerant Systems May Encounter Problems with WinNT SP3	This host is susceptible to start-up and operating failures (if using a Fault-Tolerant system) in the Clarion Agent Service. This check will verify that this software is installed and report a vulnerability if not.
28032	HF-SP3 Creating an SFM Volume on Large Partition Causes a Stop 0x24	There are 10 issues dealing with the "Services for Macintosh" (SFM) volumes that are not addressed on this host. This check will verify that this software is installed and report a vulnerability if not.
28033	HF-SP3 Denial of Service Attack Against WinNT Simple TCP/IP Services	This host is susceptible to a malicious attack against its Simple TCP/IP Service. This attack can cause increased network traffic and make the host to appear frozen; causing a Denial of Service (DOS). This check will verify that this software is installed and report a vulnerability if not.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
28034	HF-SP3 RPCSS.EXE Consumes 100% CPU due to RPC-spoofing Attack	This host is susceptible to a DOS in the Rpcss.exe process (it could consume 100 percent of CPU time) as the result of an RPC spoofing attack. This is a malicious attack on the Remote Procedure Call (RPC) service. This check will verify that this software is installed and report a vulnerability if not.
28035	HF-SP3 Denial of Service Attack Causes Windows NT Systems to Reboot	This host is susceptible to hanging during the processing of a Server Message Block (SMB) logon request; memory corruption may occur causing one of the following errors: "STOP 0x0000000A" "STOP 0x00000050" This check will verify that this software is installed and report a vulnerability if not.
28036	HF-SP3 Generic SSL (PCT/TLS) Updates for IIS and Microsoft Internet Products	Several updates to the Windows Secure Sockets Layer software are not applied on this host. This check will verify that this software is installed and report a vulnerability if not.
28037	HF-SP3 Problems Using TAPI 2.1	This host may experience one or more of the following problems while using TAPI 2.1: 1) The data an application provides for lineSetCallData is lost when Remote TSP is used. 2) TAPISRV becomes unresponsive and CPU utilization peaks at 100%. 3) TAPISRV cannot be started when RAS or another Windows NT Service starts TAPI. 4) TAPISRV causes an Access Violation error message when calling Agent functions such as lineAgentSpecific and lineGetAgentActivityList. This check will verify that this software is installed and report a vulnerability if not.
28038	HF-SP3 STOP 0x0000000A or 0x00000019 Due to Modified Teardrop Attack	This host is susceptible to hanging after receiving a number of deliberately corrupted UDP packets. This check will verify that this software is installed and report a vulnerability if not.
28039	HF-SP3 STOP 0xA Due to Buffer Overflow in NDISWAN.SYS	The host may experience a STOP 0x0000000A on a Windows NT computer when copying files via RAS over a SLIP (Serial Line Interface Protocol) connection. This check will verify that this software is installed and report a vulnerability if not.
28040	HF-SP3 Invalid UDP Frames May Cause WINS to Terminate	Invalid UDP frames directed to this host if running WINS raises an exception in WINS causing it to terminate silently. When WINS is no longer running, problems such as domain synchronization, browsing, or connectivity may occur. This check will verify that this software is installed and report a vulnerability if not.
28041	HF-SP3 "NET USER /TIMES" Command Does Not Work in Year 2000	Year 2000 issue with the NET command "/TIMES" fix not applied on this host. This check will verify that this software is installed and report a vulnerability if not.
28042	HF-SP3 User Manager Does Not Recognize February 2000 As a Leap Year	There are 13 Year 2000 (Y2K) issues that are not fixed on this host. This check will verify that this software is installed and report a vulnerability if not.
28043	HF-SP3 Using Iomega ATAPI Zip Drives with Windows NT	This host may not be able to access the disk in the ATAPI version of an Iomega Zip drive. This check will verify that this software is installed and report a vulnerability if not.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
28080	SP4 (40-bit Cipher-strength) is not installed	This check will verify that the 40-bit (exportable version) cipher-strength Service Pack 4 software is installed and report a vulnerability if not.
28081	SP4 (128-bit Cipher-strength) is not installed	This check will verify that the 128-bit (non exportable version) cipher-strength Service Pack 4 software is installed and report a vulnerability if not.
28082	SP4 is not installed	This check will verify that the Windows NT Service Pack 4 software is installed and report a vulnerability if not.
28083	HF-SP4 BIOS Date Value Does Not Immediately Update on January 1, 2000	This host is susceptible to various Year 2000 (Y2k) issues. The BIOS date/time stamp may not be immediately updated upon booting. This check will verify that this software is installed and report a vulnerability if not.
28084	HF-SP4 RRAS Computer Stops Responding to Incoming Calls Under Stress	This host is susceptible to a failure in the RRAS Service. An insufficient buffer size problem in Kmddsp.tsp may cause this host's RRAS Service to stop responding. This check will verify that this software is installed and report a vulnerability if not.
28085	HF-SP4 Executable with a Specially-Malformed Image Header May Crash Windows NT	This host is susceptible to crashing upon receipt of a malformed image header (on an executable file). This check will verify that this software is installed and report a vulnerability if not.
28086	HF-SP4 Exchange Protocols Fail After Applying Windows NT SP4	After applying Windows NT 4.0 Service Pack 4, Microsoft Exchange Internet Applications and Services may no longer function properly on this host. This check will verify that this software is installed and report a vulnerability if not.
28087	HF-SP4 WinNT 4.0 Post-Service Pack 4 Hotfixes Combined Into One Package	Several security issues are not dealt with on the host machine without the "roll-up" fixes installed. This check will verify that this software is installed and report a vulnerability if not.
28088	HF-SP4 Screen Saver Vulnerability Lets User Privileges be Elevated	The host is vulnerable to a specially designed screensaver application which could elevate the security privileges of the logged-on user. This check will verify that this software is installed and report a vulnerability if not.
28089	HF-SP4 Restricting Changes to Base System Objects	The host computer is vulnerable to a malicious locally logged-in user to elevate his privilege to Administrator. This check will verify that this software is installed and report a vulnerability if not.
28090	HF-SP4 MSMQ Err: Error While Creating MSMQ Internal Certificate	When you click Renew Internal Certificate in the Microsoft Message Queue Control Panel tool on February 29 of a leap year (for example, the year 2000, 2004, 2008, and so on), the following error message is displayed: "Error while creating MSMQ internal certificate. Error: 0x8000ffff" This check will verify that this software is installed and report a vulnerability if not.
28091	HF-SP4 "NET USER /TIMES" Command Does Not Work in Year 2000	Year 2000 issue with the NET command "/TIMES" fix not applied on this host. This check will verify that this software is installed and report a vulnerability if not.
28092	HF-SP4 WinNT Lets You Paste Text into Unlock Workstation Dialog Box	This host is susceptible to a malicious user accessing the first line of clipboard-text from the locked console. This check will verify that this software is installed and report a vulnerability if not.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
28150	SP5 (40-bit Cipher-strength) is not installed	This check will verify that the 40-bit (exportable version) cipher-strength Service Pack 5 software is installed and report a vulnerability if not.
28151	SP5 (128-bit Cipher-strength) is not installed	This check will verify that the 128-bit (non exportable version) cipher-strength Service Pack 5 software is installed and report a vulnerability if not.
28152	SP5 is not installed	This check will verify that the Service Pack 5 software is installed and report a vulnerability if not.
28153	HF-SP5 Exceeding MaxRequestThreads May Cause Windows NT to Hang	This host is susceptible to a DOS attack by a malicious service process running locally. This check will verify that this software is installed and report a vulnerability if not.
28154	HF-SP5 "Access Violation" Error Message When You Quit Phone Dialer	When this host quits the Phone Dialer, it may receive an "Access violation" error message. This check will verify that this software is installed and report a vulnerability if not.
28155	HF-SP5 Malformed IGMP Packets May Promote "Denial of Service" Attack	This host is susceptible to a DOS attack as a fragmented IGMP packet may cause the TCP/IP stack to improperly gain access to invalid segments of the computer's memory. This can the degrade the host's performance until it stops responding (hangs). This check will verify that this software is installed and report a vulnerability if not.
28156	HF-SP5 Denial of Service Attack Using Unprotected IOCTL Function Call	A rogue program running on this host making certain IOCTL Device calls may cause the host to be in a DOS situation (with the mouse and keyboard). This check will verify that this software is installed and report a vulnerability if not.
28157	HF-SP5 Malformed Request Causes LSA Service to Hang	A specially malformed request to the Microsoft Local Security Authority (LSA) service may be used to exploit a security vulnerability on this host. A user can abuse this vulnerability to run a program and cause a denial of service attack that may cause the LSA service to stop responding (hang) and require a restart of the host. This check will verify that this software is installed and report a vulnerability if not.
28158	HF-SP5 NETDDE.EXE Fails to Relay WM_DDE_TERMINATE to Remote Clients	This host is vulnerable to having applications "orphaned" by a Network DDE call to terminate the application. This check will verify that this software is installed and report a vulnerability if not.
28159	HF-SP5 Memory Leak When Performance Counters Are Not Available	When a program that attempts to gain access to a performance counter that has not been installed, this host's performance may degrade or may stop responding (hang) because of a memory leak. This check will verify that this software is installed and report a vulnerability if not.
28160	HF-SP5 File Corruption on an NTFS Volume with More Than 4 Million Files	This host is susceptible to disk corruption on NTFS volumes. This check will verify that this software is installed and report a vulnerability if not.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
28161	HF-SP5 Malformed Phonebook Entry Security Vulnerability in RAS Client	A specially malformed phonebook entry may be used to exploit a security vulnerability on a computer that has the Microsoft Remote Access Service (RAS) client software installed. A user with the proper permissions can abuse this vulnerability to run a program and cause a denial of service attack or privilege escalation attack on the RAS client computer. This check will verify that this software is installed and report a vulnerability if not.
28162	HF-SP5 DUN Credentials Cached When Save Password Not Selected with RAS	This host is susceptible to having its client's passwords (from the Dial-Up Networking Client) cached on the disk regardless of application settings. This applies to the Remote Access Service (RAS). This check will verify that this software is installed and report a vulnerability if not.
28163	HF-SP5 Exchange Clients Appear to Intermittently Hang During Normal Operation	This host's Exchange Server may be vulnerable to a problem servicing slower Exchange and Outlook Clients. The clients may appear to stop responding (as they are waiting for higher-speed clients to complete their exchange of datagrams. This check will verify that this software is installed and report a vulnerability if not.
28164	HF-SP5 DUN Credentials Cached When Save Password Not Selected with RRAS	This host is susceptible to having its client's passwords (from the Dial-Up Networking Client) cached on the disk regardless of application settings. This applies to the Routing and Remote Access Service (RRAS). This check will verify that this software is installed and report a vulnerability if not.
28165	HF-SP5 Fix for IP Source Routing Vulnerability	The host computer is vulnerable to security breaches in the TCP/IP Routing area. This check will verify that this software is installed and report a vulnerability if not.
28166	HF-SP5 Malformed Help File Causes Help Utility to Stop Responding	This host is vulnerable to a specially -malformed Microsoft Help file that is used to exploit a security vulnerability. When a user activates the Windows Help file tool (for example, by pressing the F1 key) this vulnerability may be used to run a malicious program and may cause the Help file tool to stop responding (hang). This check will verify that this software is installed and report a vulnerability if not.
28167	HF-SP5 BIOS Date Value Does Not Immediately Update on January 1, 2000	This host is susceptible to date-rollover problems (at century rollover) in some older BIOS's. This check will verify that this software is installed and report a vulnerability if not.
28168	HF-SP5 XIMS: NNTP Service Converts Two Digit Years Incorrectly	This host's Network News Transfer Protocol (NNTP) service may not properly convert two-digit years to four digits. This check will verify that this software is installed and report a vulnerability if not.
28169	HF-SP5 "NET USER /TIMES" Command Does Not Work in Year 2000	Year 2000 issue with the NET command "/TIMES" fix not applied on this host. This check will verify that this software is installed and report a vulnerability if not.
28173	SP6 (40-bit Cipher-strength) is not installed	This check will verify that the 40-bit (non exportable version) cipher-strength Service Pack 6 software is installed and report a vulnerability if not.
28174	SP6 (128-bit Cipher-strength) is not installed	This check will verify that the 128-bit (non exportable version) cipher-strength Service Pack 6 software is installed and report a vulnerability if not.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
28175	SP6 is not installed	This check will verify that the Service Pack 6 software is installed and report a vulnerability if not.
28176	HF-SP6 Security Descriptor Allows Privilege Elevation on Remote Computers	A malicious user may be able to cause a different program to run in place of Rasman. Significantly, this program would run in the System context and allow the program to take almost any action on the computer. This check will verify that this software is installed and report a vulnerability if not.
28200	Secure Channel SSL 40-bit Cipher-strength not applied	This check will verify that the 40-bit Secure Channel SSL (Secure Sockets Layer) software is installed and report a vulnerability if not.
28201	Secure Channel SSL 128-bit Cipher-strength not applied	The security policy indicates that the 128-bit Secure Channel SSL (Secure Sockets Layer) software should be installed. This check will verify that this software is installed and report a vulnerability if not.
28250	HF-WWW Page Contents Visible When Certain Characters are at End of URL	The IIS Server may be susceptible to exposing the internal contents of its scripts. Only foreign versions and the English version with the Far East Language Pack are susceptible. This check will verify that this software is installed and report a vulnerability if not. Note that this check is for a "hotfix", therefore it will return vulnerable even if the applicable WWW service is not installed.
28251	HF-WWW Specially-Malformed FTP Requests May Create Denial of Service	Specially-malformed FTP requests may create a Denial of Service in the FTP service, which causes Internet Information Server (IIS) to stop responding and generate an Access Violation error message. This check will verify that this software is installed and report a vulnerability if not. Note that this check is for a "hotfix", therefore it will return vulnerable even if the applicable WWW service is not installed.
28252	HF-WWW Specially-Malformed Header in GET Request Creates Denial of Service	A specially-malformed header in a GET request can create a Denial of Service in the W3 server and use all available memory on the Web server, causing Internet Information Server (IIS) to stop responding to any request. This check will verify that this software is installed and report a vulnerability if not. Note that this check is for a "hotfix", therefore it will return vulnerable even if the applicable WWW service is not installed.
28253	HF-WWW NTFS Alternate Data Stream Name of a File May Return Source	This host may be susceptible to allowing the script source for a web page to be viewed. This check will verify that this software is installed and report a vulnerability if not. Note that this check is for a "hotfix", therefore it will return vulnerable even if the applicable WWW service is not installed.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
28254	HF-WWW FTP Passive Mode May Terminate Session	The Internet Information Server FTP service includes a passive mode command (PASV) to request that the server wait for a connection instead of initiating one after receiving a transfer command. Certain situations using multiple passive connections may result in errors, problems with system performance as well as denial of service situations for both the Web and FTP services. This check will verify that this software is installed and report a vulnerability if not. Note that this check is for a "hotfix", therefore it will return vulnerable even if the applicable WWW service is not installed.
28255	HF-WWW Specially-Malformed GET Requests Can Create Denial of Service	FTP Get Commands may cause a DOS against the IIS Server on the host. This check will verify that this software is installed and report a vulnerability if not. Note that this check is for a "hotfix", therefore it will return vulnerable even if the applicable WWW service is not installed.
28256	HF-WWW Settings May Not Be Applied with URL with Short Filename	Some configuration settings on the IIS Server may not be applied on this host. This check will verify that this software is installed and report a vulnerability if not. Note that this check is for a "hotfix", therefore it will return vulnerable even if the applicable WWW service is not installed.
29000	Windows NT - Third Party Software	
29001	Outdated Version of Netscape Communicator	The target host was found to be running an outdated version of the Netscape WWW browser. This module checks specifically for versions less than version 4.61.
29002	SLMail insecure registry permissions	The target host was found to have insecure registry permissions set on the Seattle Labs SLMail configuration key.
29003	IIS 2.0/3.0 Installed	The target host was found to be running IIS version 2.0 or 3.0. IIS version 2.0/3.0 was known to contain a number of security problems which are fixed in newer versions.
29008	Insecure logon method allowed for MS IIS Web Server	The target host was found configured to allow MS Peer Web Server (IIS/FTP/Gopher) logon connections that are unsecured. If the target host is using Batch mode it could be exposing other network resources from a remote Web Browser. If the host is using Network mode it also can be remotely accessed by a browser but it will not share resources with the remote machine.
29009	Insecure logon method allowed for MS IIS FTP service	The target host was found to allow unsecured logon modes to the Microsoft Internet Information Server's FTP service. Local User logon should be set for the most secure operation of the Server.
29010	Insecure logon method allowed for MS IIS Gopher service	The target host was found to allow unsecured logon modes to the Microsoft Internet Information Server's FTP service. Local User logon should be set for the most secure operation of the Server.
29011	IIS Anonymous FTP access permitted	The target host was found to have anonymous FTP access enabled. Anonymous users are permitted to connect to an IIS FTP server by default, however your security policy indicates that access should be restricted to authenticated users only.

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
29012	IIS Anonymous Gopher access permitted	By default, Microsoft's IIS Gopher server allows anonymous users to connect and access available information. While this may be the desired configuration, if restricted or sensitive information is available via the Gopher server, access should be limited to authenticated users only.
29013	IIS WWW Guest access permitted	The target host was found to have Guest WWW access enabled. This allows the Guest network user to connect to the IIS WWW server on the target host.
29014	IIS WWW Special characters permitted	The target host was found to be configured to allow special characters to be passed to shell commands. The security policy indicates that this should not be permitted.
29015	IIS WWW CreateProcess enabled	The target host was found to be configured to run CGI scripts in the system context instead of the IIS IUSR_ user. The security policy indicates that this should not be permitted.
29016	IIS WWW Successful logging disabled	The target host was found to have the logging of successful HTTP requests disabled. The security policy indicates that this logging should be enabled.
29017	IIS WWW Error logging disabled	The target host was found to have the logging of erroneous HTTP requests disabled. The security policy indicates that this logging should be enabled.
29018	IIS WWW Server Side Includes	The target host was found to have server side include functionality enabled. The security policy specifies that this functionality should be disabled.
29019	IIS FTP Guest Access Permitted	The target host's FTP service was found to be configured to allow GUEST access. The security policy indicates that GUEST access should be disabled.
29021	IIS FTP bounce attack enabled	The target host's FTP service was found to have the FTP bounce attack enabled. The security policy indicates that this option should be disabled.
29022	IIS FTP anonymous usage logging disabled	The target host's FTP service was found to have the logging of anonymous access disabled.
29023	IIS FTP regular user usage logging disabled	The target host's FTP service was found to have the logging of regular user access.
30000	Windows NT - Services	
30001	Unrecognized Service found	An unrecognized Service was detected on the target host.
30002	Service found logged-on under a User Account	A service was found to be running (logged-in) as a User on the target host. Most Windows NT Services run in the "System Account". Briefly, services running in the System Account (remember that services are started before the login procedure) have very limited access to remote resources. They essentially are given the "Everyone" account's permissions, which are (hopefully) very restricted.
30003	Alerter Service detected	The Windows NT Alerter service was found to be running on the target host. The Alerter is used to forward alerts generated on the local host to remote computers or user names.
30004	Messenger Service detected	The messenger service was found to be running on the target host. The Messenger Service is used to exchange short messages between Users (that are running the Service).

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
30005	Messenger Service Found and a Popup-Message was Sent to Host	The messenger service was found to be running on the target host and a Windows Popup message has been sent to the target.
30006	Remote Access Service detected	The Remote Access Service (RAS) was detected on the target host. RAS lets remote users dial into a Windows NT RAS server and use the resources of its network as if directly connected. In its simplest mode, users logging on to Windows NT remotely simply check a small box on their logon window that automatically establishes the RAS connection and authenticates the session.
30007	Network Monitor Service detected	The Network Monitor service was detected on the target host. Network Monitor is a network diagnostic tool that monitors local area networks and provides a graphical display of network statistics. Network administrators can use these statistics to perform routine trouble-shooting tasks, such as locating a server that is down, or that is receiving a disproportionate number of work requests .
30008	PC Anywhere Service detected	The PC Anywhere service was detected on the target host. This Symantec product is a "remote desktop", used to provide a remote user with a virtual desktop of this host.
30009	Remote Desktop Service detected	The Remote Desktop service was detected on the target host. This NAI product is a "remote desktop", used to provide a remote user with a virtual desktop of this host.
30010	Simple TCP/IP Service detected	The Simple TCP/IP Service was detected on the target host.
30012	Host set to suppress Interactive Services	The Target host was found configured to disallow Services from interacting with the logged-on User. This will likely disable Security Services from gaining a password after log-in.
31000	Windows NT - Remote Access Server	
31001	Maximum number of allowable log-in attempt retries not set to default value	The target Server was found to have a suspicious RAS setting allowing more than the default number unsuccessful tries at remote log-in.
31002	Maximum time limit for authentication not set to default value	The target Server was found to have a suspicious RAS setting allowing too much time for a remote log-in.
31003	No time limit on connections - inactive users will never be disconnected	The target Server was found to have the RAS/NetBIOS Gateway setting of AutoDisconnect set to a non -default value. This means a dialed-in User may never be purposely disconnected even after prolonged periods of inactivity. You should review this setting in accordance to your security policy.
31004	Broadcast Datagrams are being forwarded to Remote hosts	The target Server was found to have RAS/NetBIOS Gateway settings that forward Broadcast Datagram packets to the remote host. Though this may be a security concern it is more likely a poor performance choice (unless the remote host absolutely needs to see the Broadcast Datagram packets on the network).

Table D.1: The CyberCop Scanner vulnerability database (continued)

Vuln.ID	Vulnerability name	Vulnerability description
31005	Auditing is turned off (Event/Security log will not contain RAS events)	The target Server was found to have a suspicious RAS setting disabling the RAS Server's ability to perform auditing.
31006	Authentication test-password sent in Clear Text	The target Server was found to have RRAS (Routing and Remote Access Server)/PPP Gateway settings that allow the Authentication password (sent during CHAP) to be passed in Clear Text. It is possible it may also be configured to not authenticate at all.
31007	Maximum number of Config-Reject packets not set to default value	The target Server was found to have RAS/PPP Gateway settings that allow more than the default number of Config-Reject packets to be sent before the PPP client is deemed to be not able to connect. A higher number may legitimately be used for some Unix PPP Clients.
31008	Maximum number of CNAK packets not set to default value	The target Server was found to have RAS/PPP Gateway settings that allow more than the default number of Configuration Negative Acknowledgments before deciding the authentication is not converging.
31009	Maximum number of unanswered Configure-Request packets not set to default value	The target Server was found to have RAS/PPP Gateway settings that indicates the number of Configure-Request packets - sent without receiving a valid Configure-Ack, Configure-Nak, or Configure-Reject, before assuming that the peer is unable to respond - is set to a non-default value.
31010	Maximum number of unanswered Terminate-Request packets not set to default value	The target Server was found to have RAS/PPP Gateway settings that allowed more than the default number of Terminate-Requests - without receiving a Terminate-Ack packet - to be sent before determining that the connection is not converging.
31011	NBGateway - Suspicious priority to Multicast Datagram packets	The target Server was found to have RAS/NetBIOS set for Multicast Datagram packets to have priority over regular NetBIOS Session traffic.
31012	NBGateway - NetBIOS Session auditing turned off	The target Server was found to have RAS/NetBIOS Gateway settings that have the NetBIOS auditing log turned off. You may want to track the resources accessed during a remote host's RAS session to track suspicious activity. This is a C2 compliant setting.

APPENDIX E

VULNERABILITY HISTORY DATA

In order to test the VF Prototype, 15 sets of history scan data were collected. Each of these sets of history scan data are summarised in figure E.1 to figure E.15 respectively. Figure E.16 shows the history scan data for the scan that was conducted after a vulnerability forecast has been done. Each figure shows the number of matched vulnerabilities uncovered for each of the CyberCop Scanner vulnerability categories over the 59 hosts in the scan scenario as explained in chapter 8.

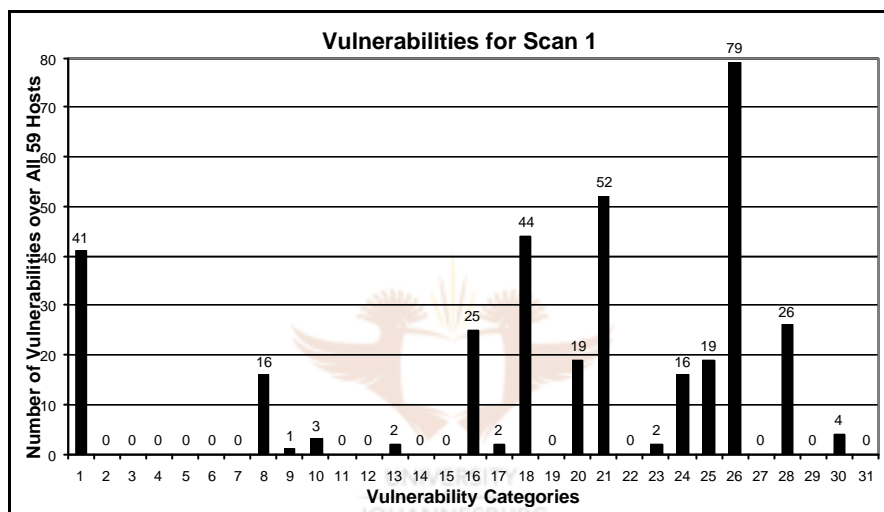


Figure E.1: Vulnerability history scan data – scan 1

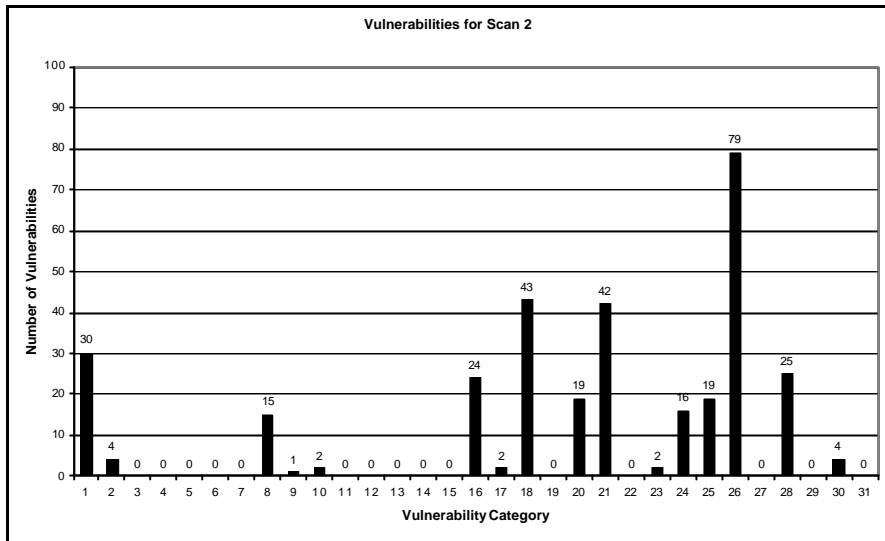


Figure E.2: Vulnerability history scan data – scan 2

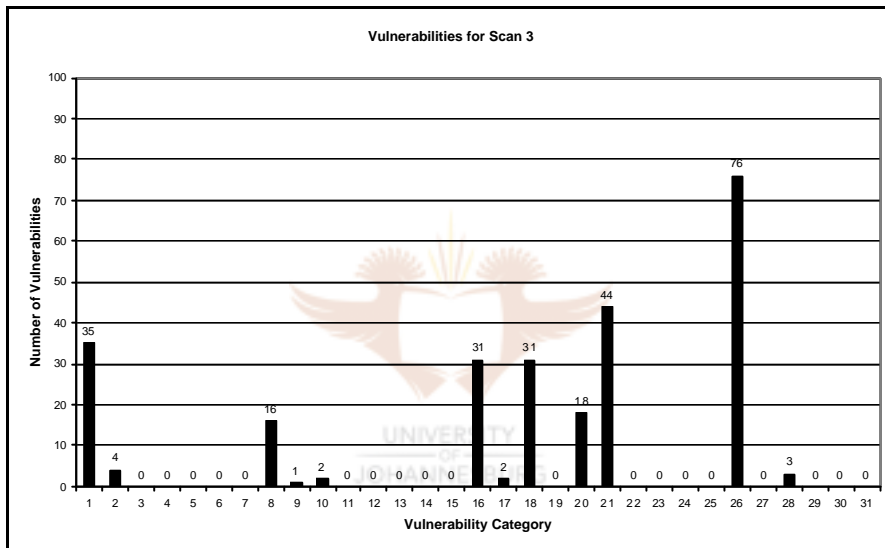


Figure E.3: Vulnerability history scan data – scan 3

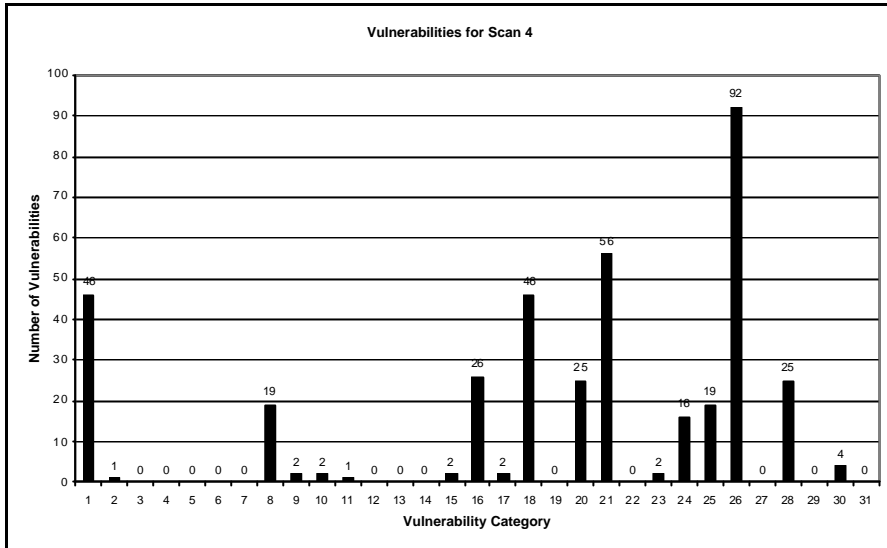


Figure E.4: Vulnerability history scan data – scan 4

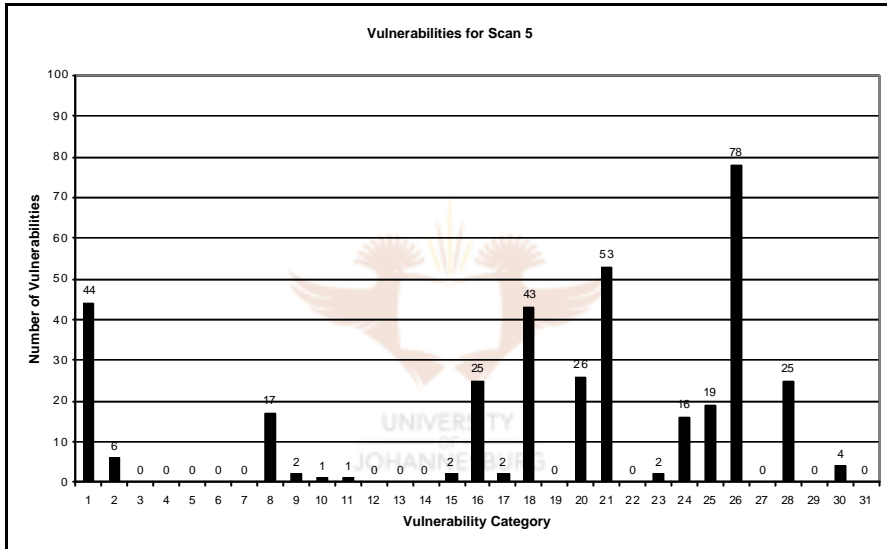


Figure E.5: Vulnerability history scan data – scan 5

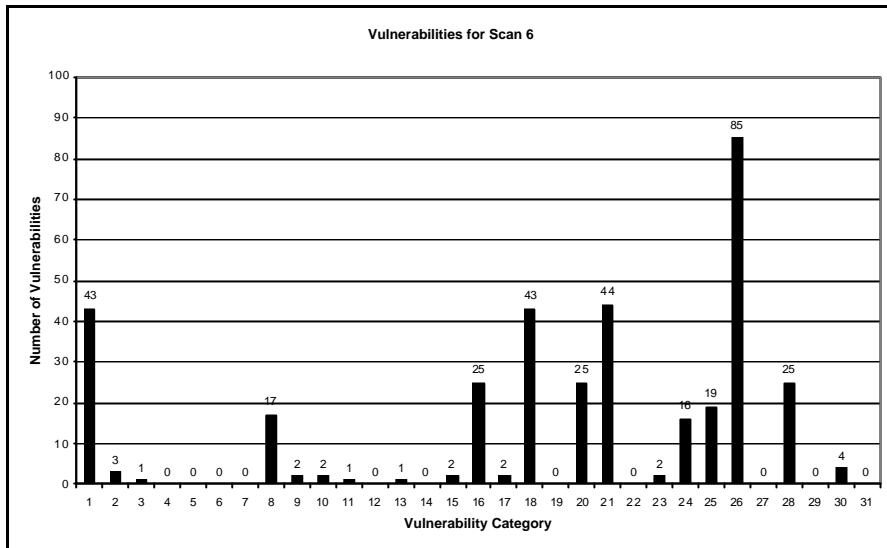


Figure E.6: Vulnerability history scan data – scan 6

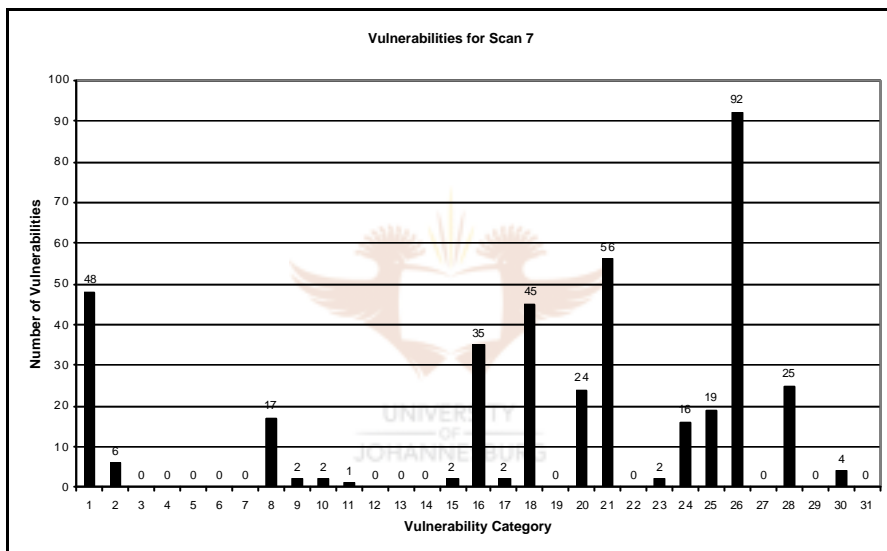


Figure E.7: Vulnerability history scan data – scan 7

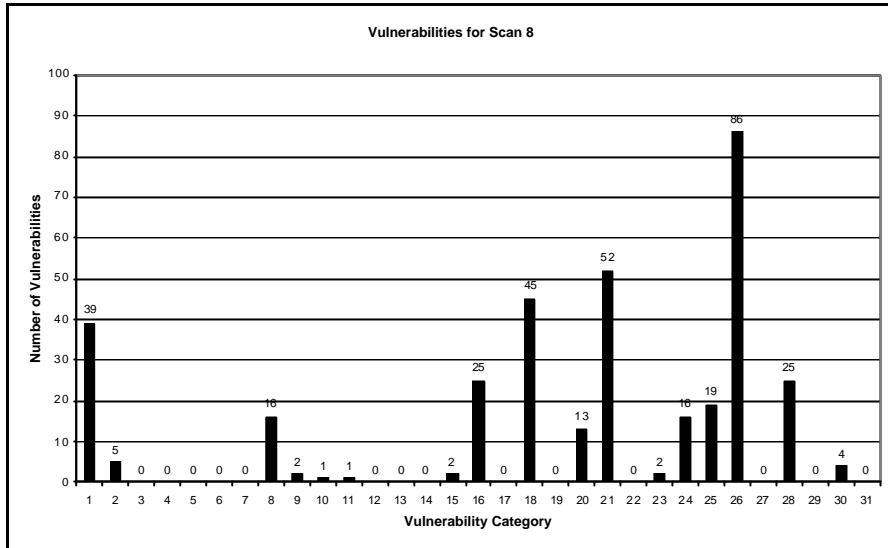


Figure E.8: Vulnerability history scan data – scan 8

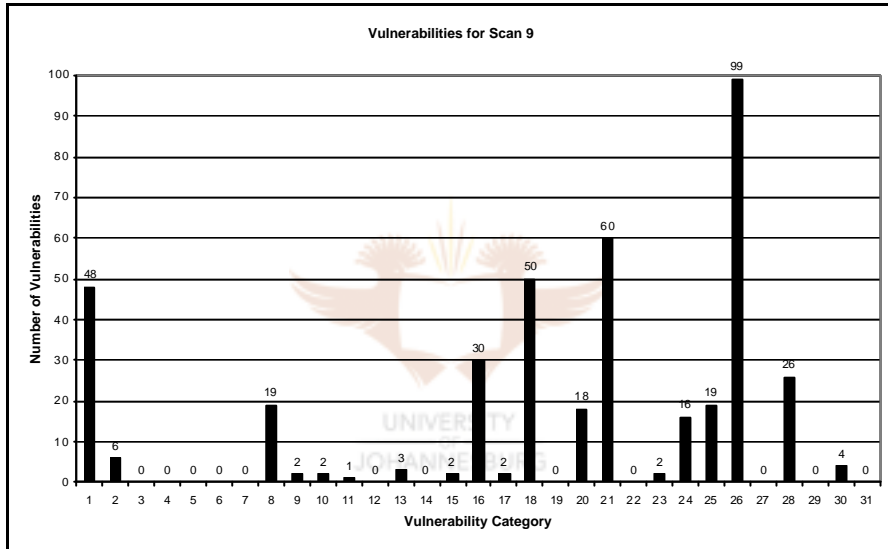


Figure E.9: Vulnerability history scan data – scan 9

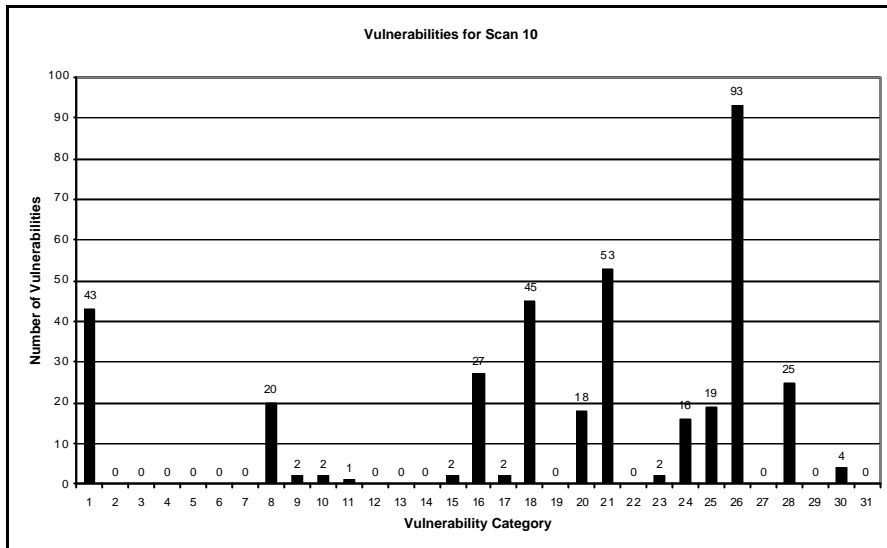


Figure E.10: Vulnerability history scan data – scan 10

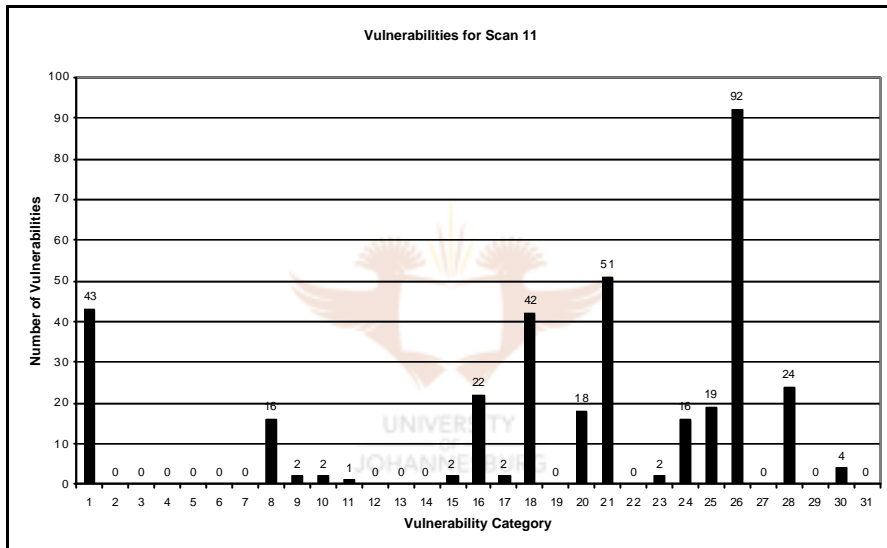


Figure E.11: Vulnerability history scan data – scan 11

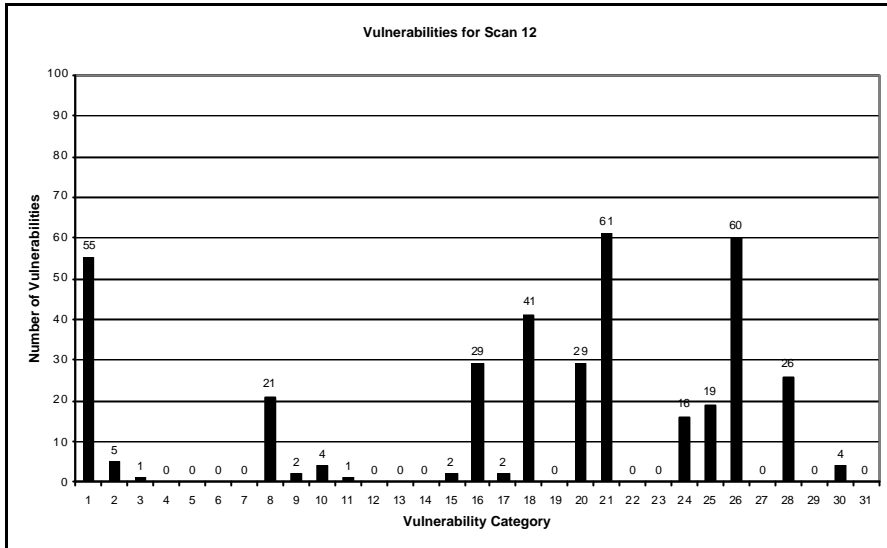


Figure E.12: Vulnerability history scan data – scan 12

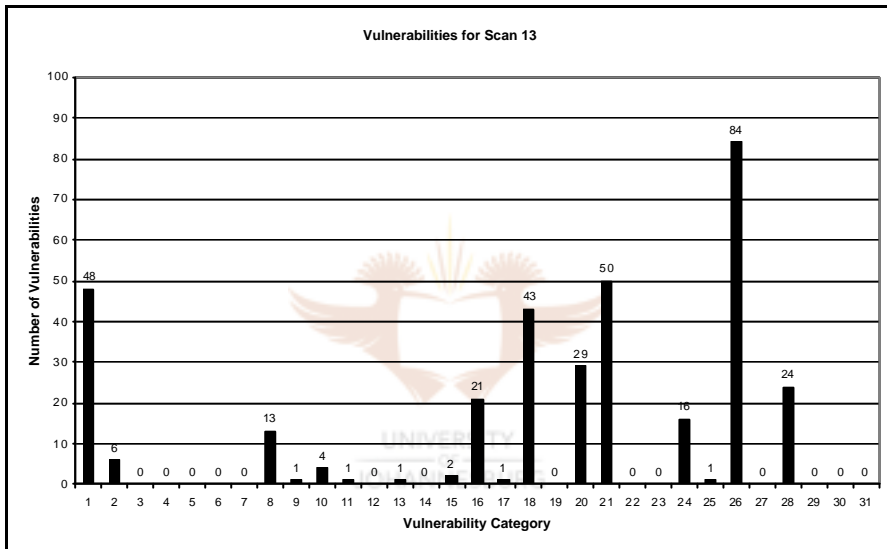


Figure E.13: Vulnerability history scan data – scan 13

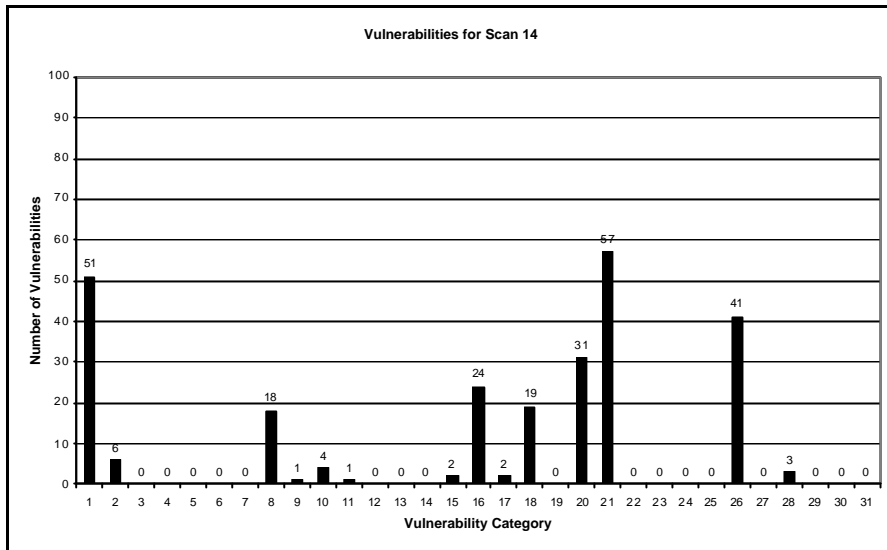


Figure E.14: Vulnerability history scan data – scan 14

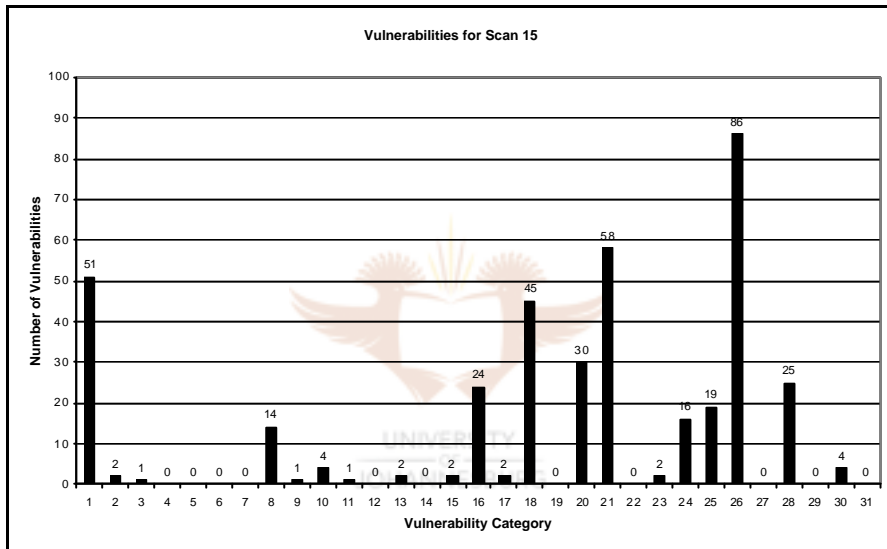


Figure E.15: Vulnerability history scan data – scan 15

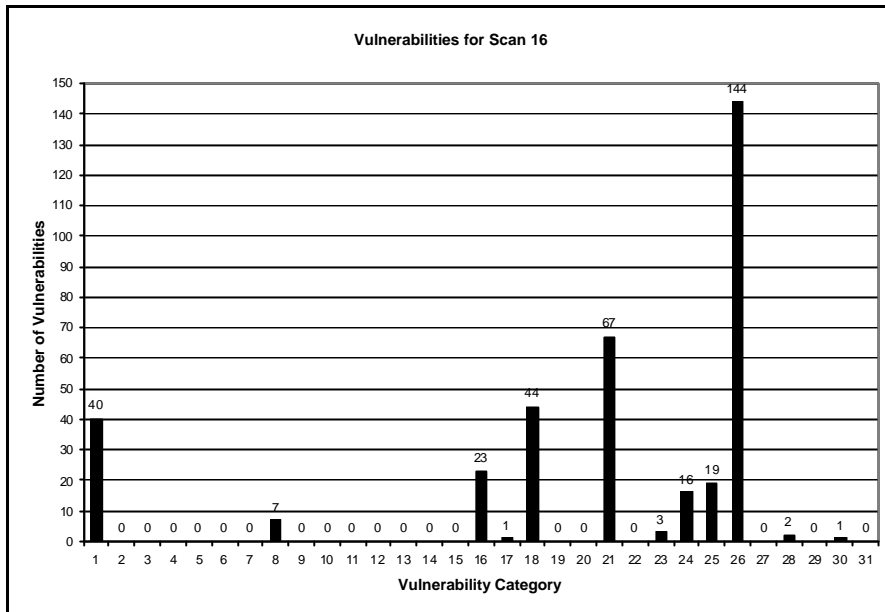


Figure E.16: Vulnerability history scan data – scan 16

In the figures above, the data is shown for each scan conducted. In order to compare the scan data according to each CyberCop Scanner vulnerability category, the data for all scans are compared in each of figures E.17 to E.48, each time for a specific CyberCop Scanner vulnerability category.



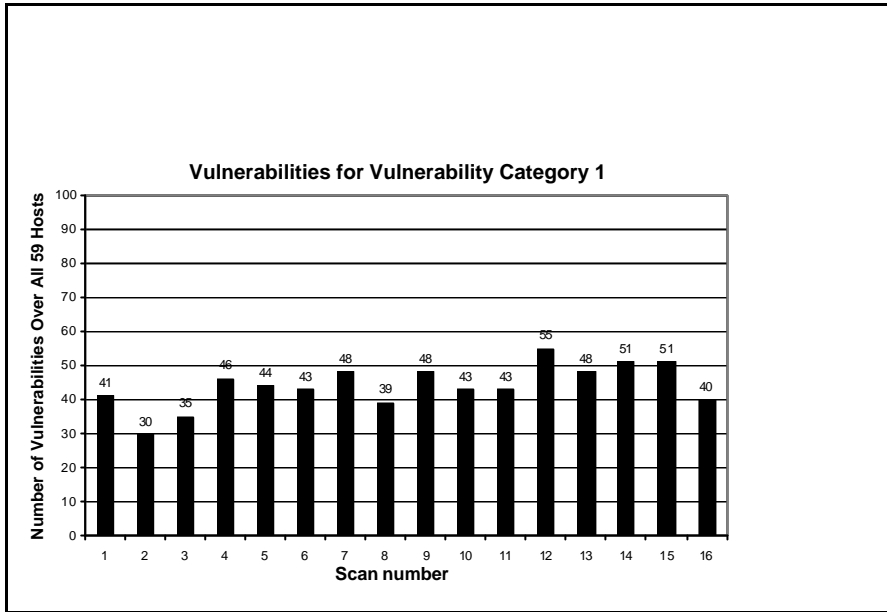


Figure E.17: Scan results over the 16 scans for CyberCop vulnerability category 1

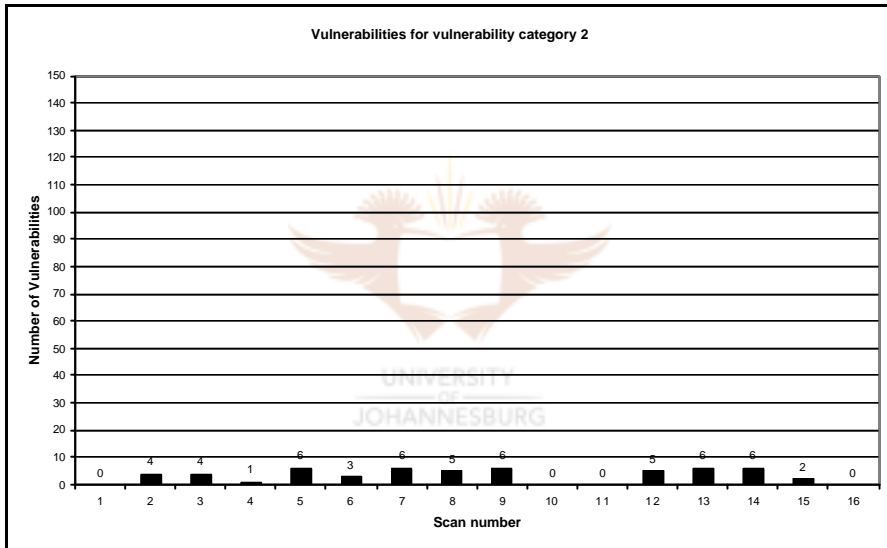


Figure E.18: Scan results over the 16 scans for CyberCop vulnerability category 2

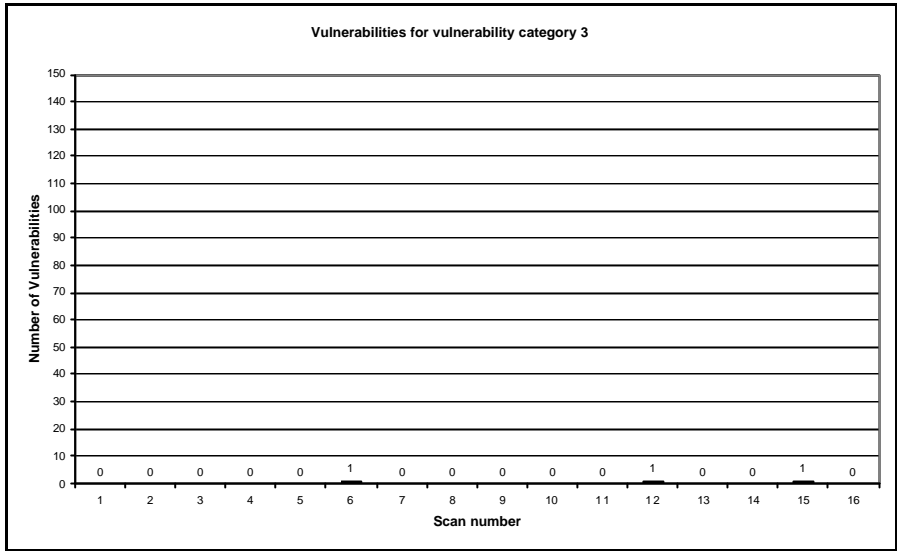


Figure E.19: Scan results over the 16 scans for CyberCop vulnerability category 3

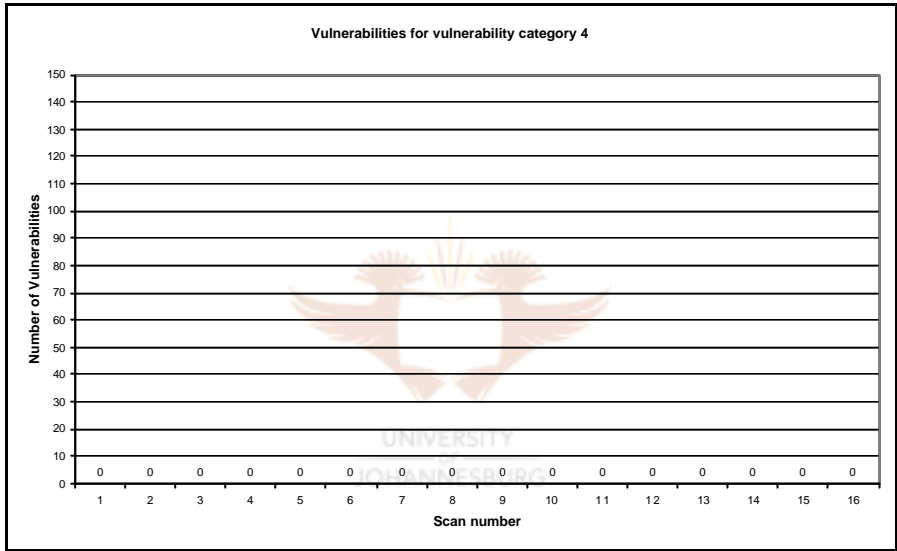


Figure E.20: Scan results over the 16 scans for CyberCop vulnerability category 4

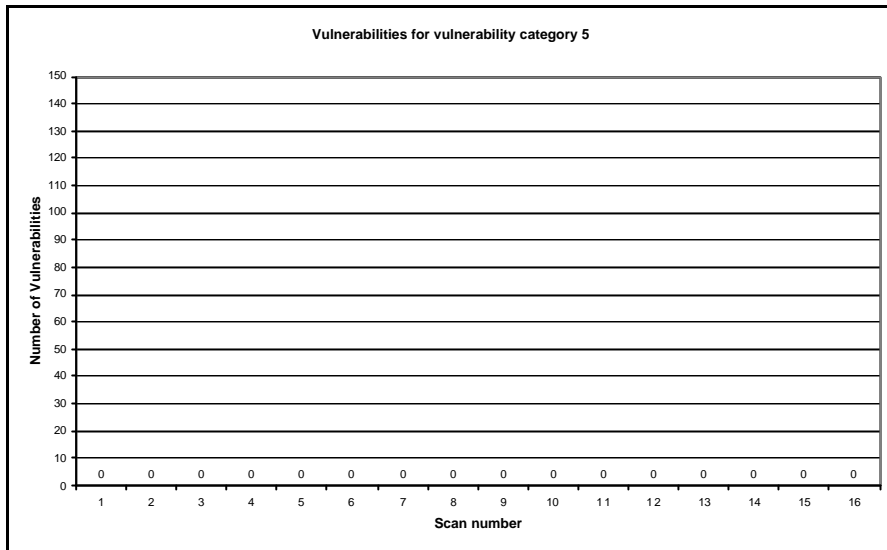


Figure E.21: Scan results over the 16 scans for CyberCop vulnerability category 5

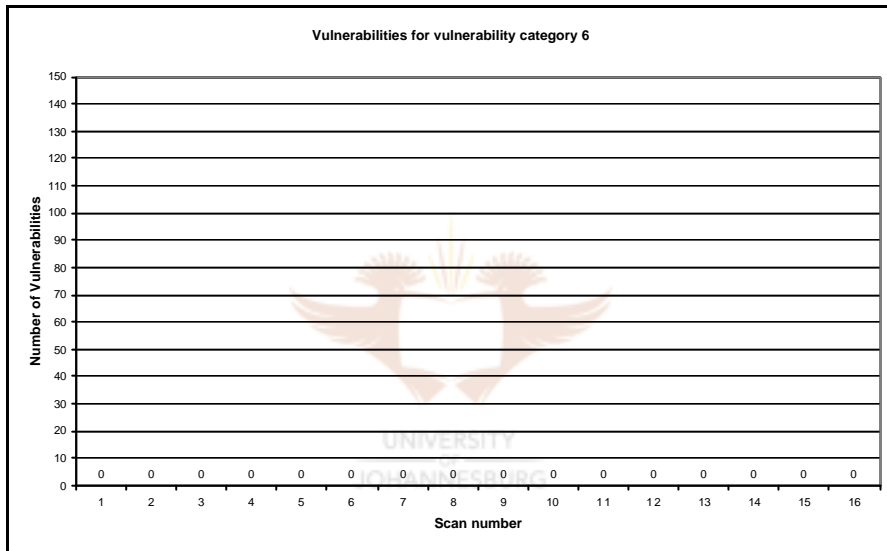


Figure E.22: Scan results over the 16 scans for CyberCop vulnerability category 6

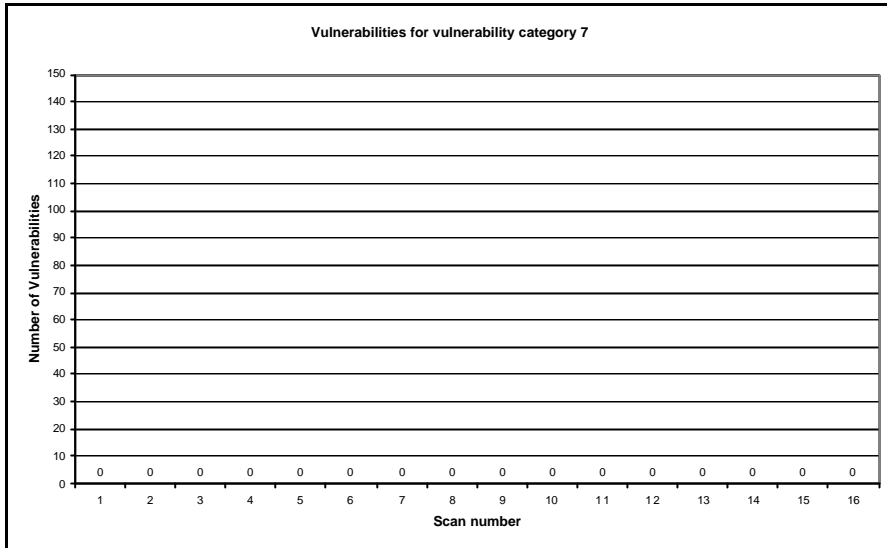


Figure E.23: Scan results over the 16 scans for CyberCop vulnerability category 7

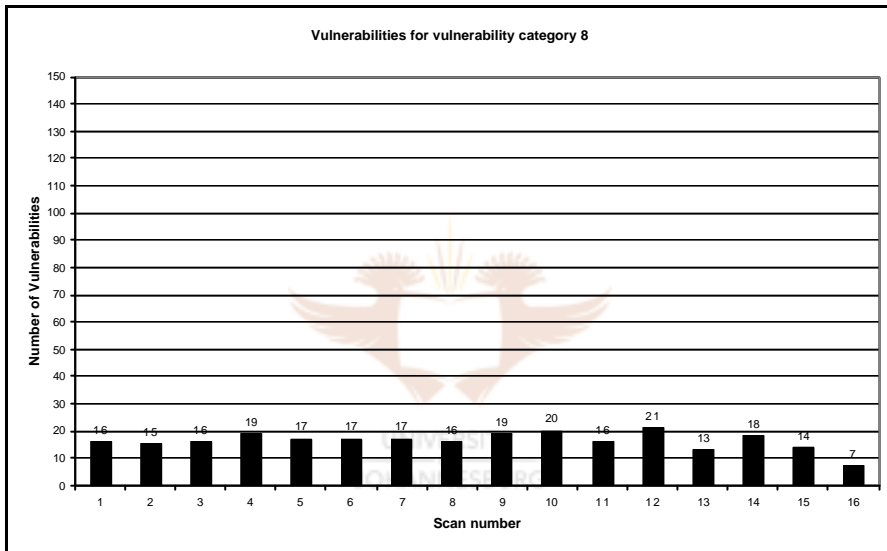


Figure E.24: Scan results over the 16 scans for CyberCop vulnerability category 8

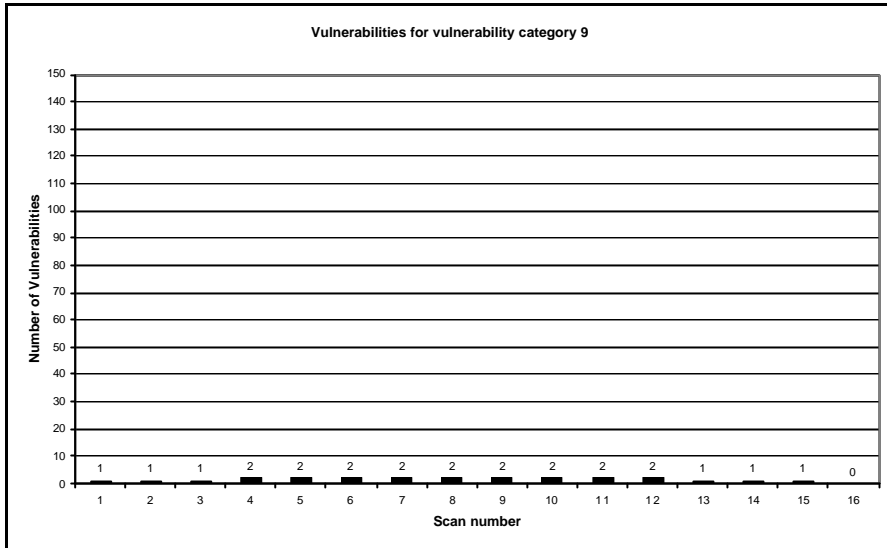


Figure E.25 Scan results over the 16 scans for CyberCop vulnerability category 9

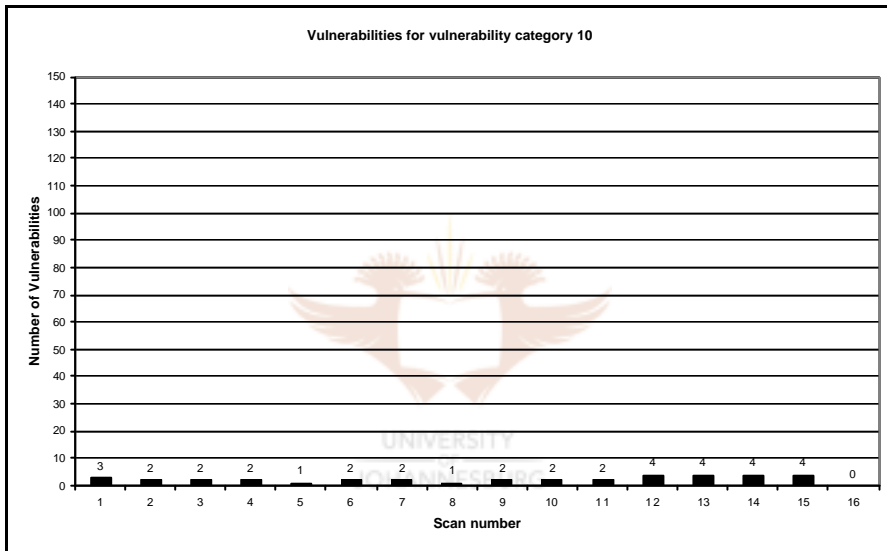


Figure E.26: Scan results over the 16 scans for CyberCop vulnerability category 10

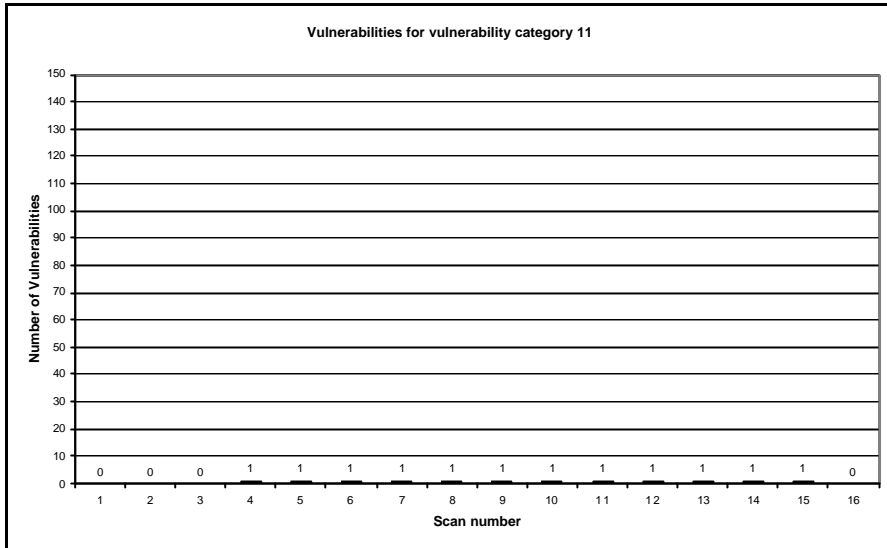


Figure E.27: Scan results over the 16 scans for CyberCop vulnerability category 11

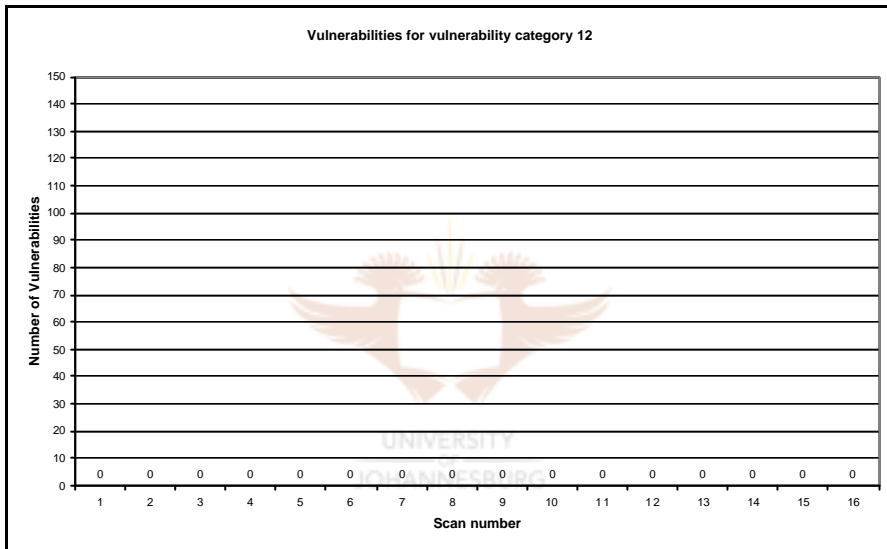


Figure E.28: Scan results over the 16 scans for CyberCop vulnerability category 12

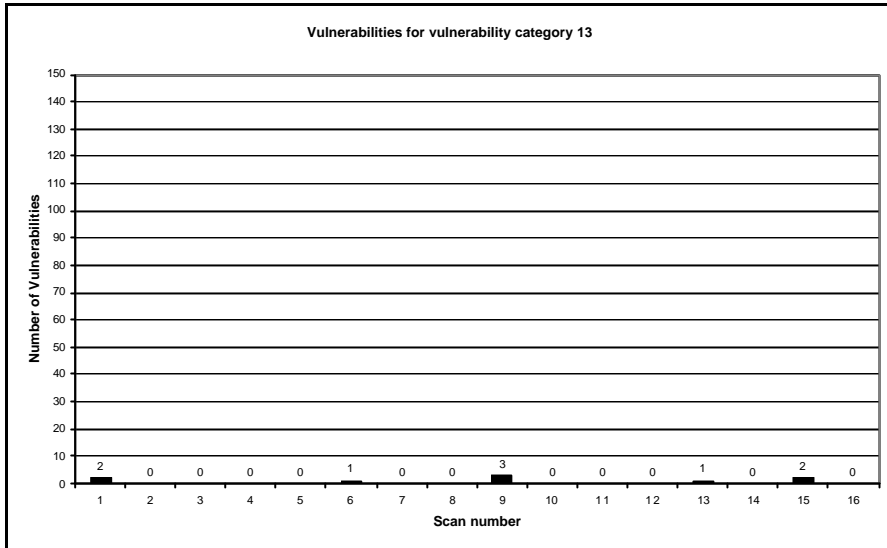


Figure E.29: Scan results over the 16 scans for CyberCop vulnerability category 13

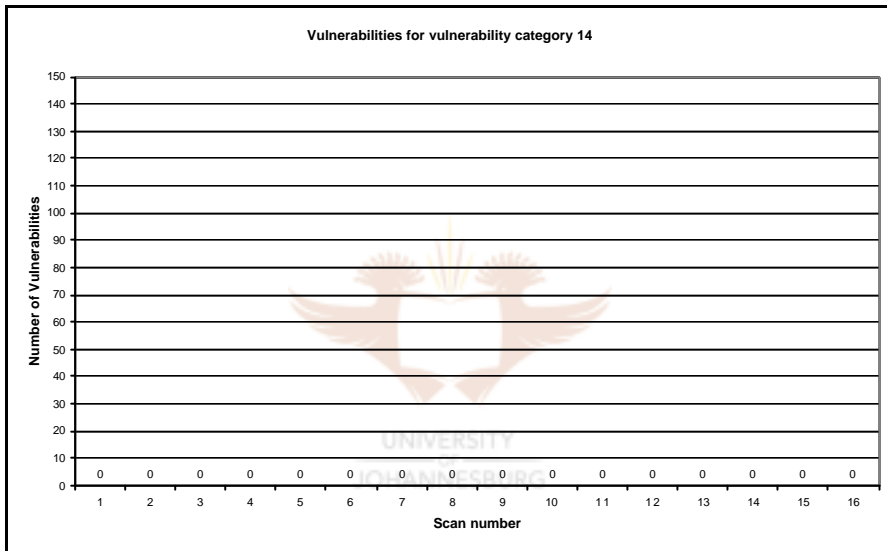


Figure E.30: Scan results over the 16 scans for CyberCop vulnerability category 14

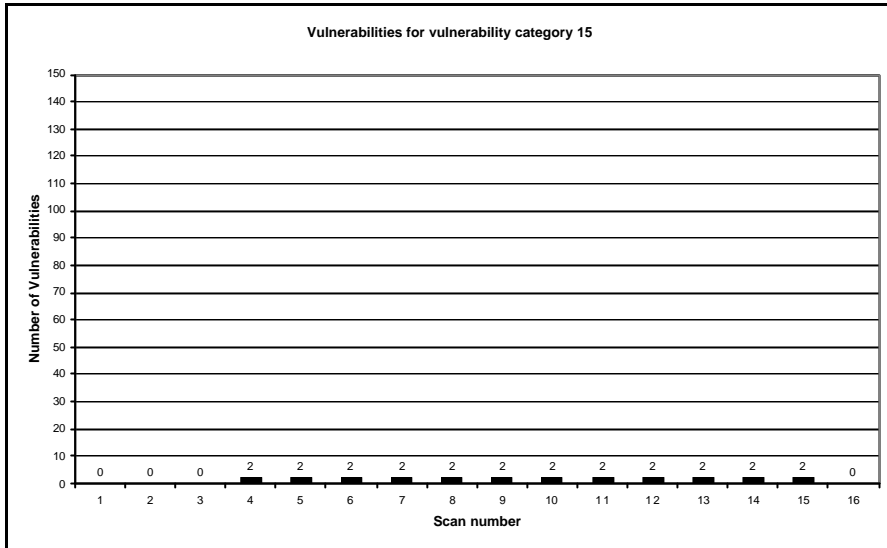


Figure E.31: Scan results over the 16 scans for CyberCop vulnerability category 15

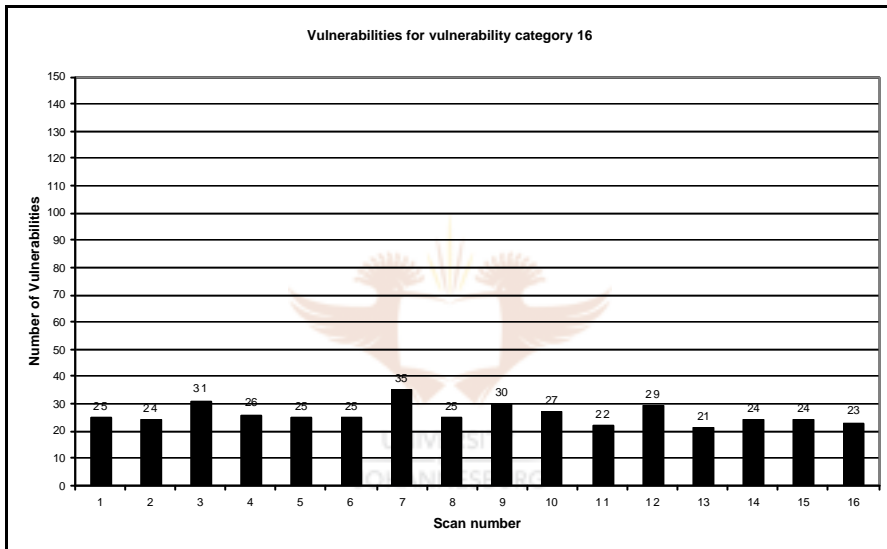


Figure E.32: Scan results over the 16 scans for CyberCop vulnerability category 16

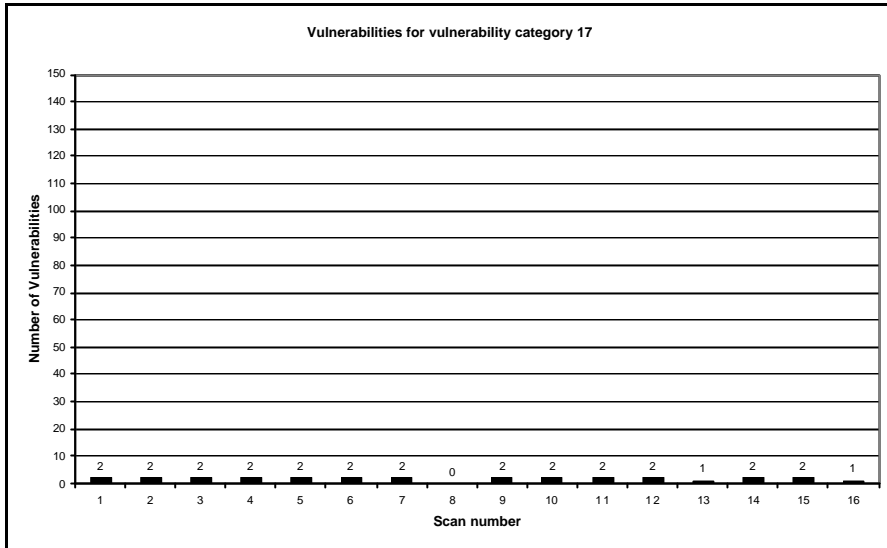


Figure E.33: Scan results over the 16 scans for CyberCop vulnerability category 17

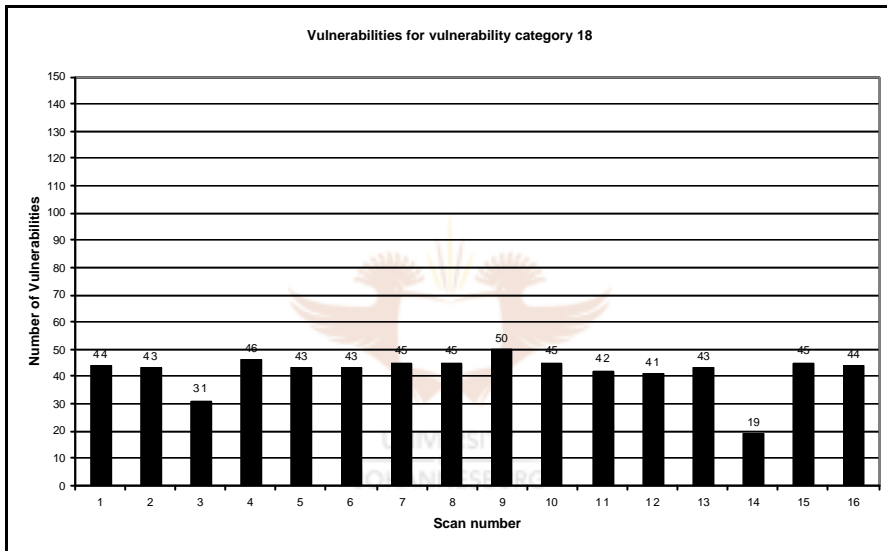


Figure E.34: Scan results over the 16 scans for CyberCop vulnerability category 18

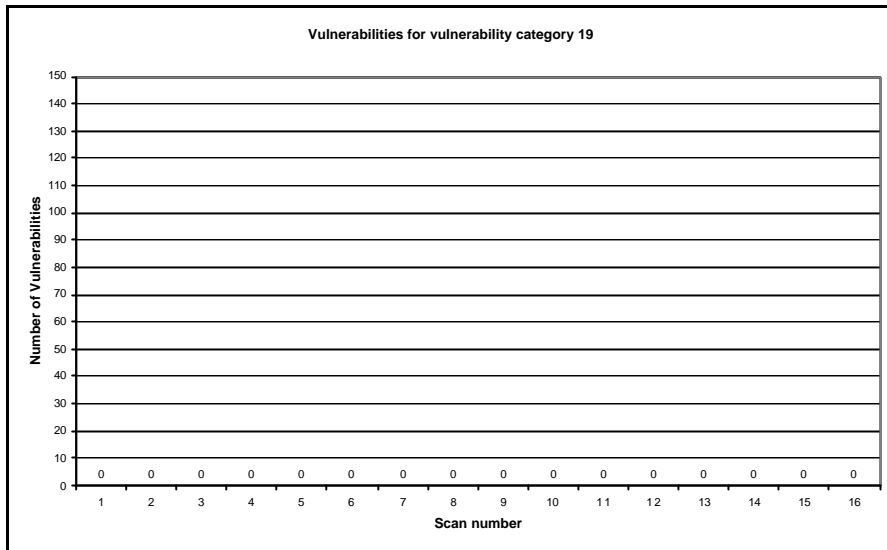


Figure E.35: Scan results over the 16 scans for CyberCop vulnerability category 19

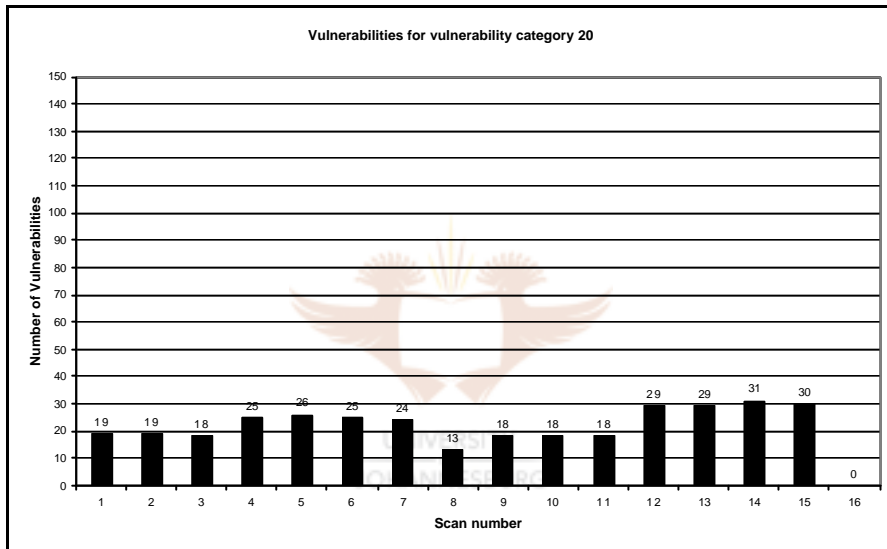


Figure E.36: Scan results over the 16 scans for CyberCop vulnerability category 20

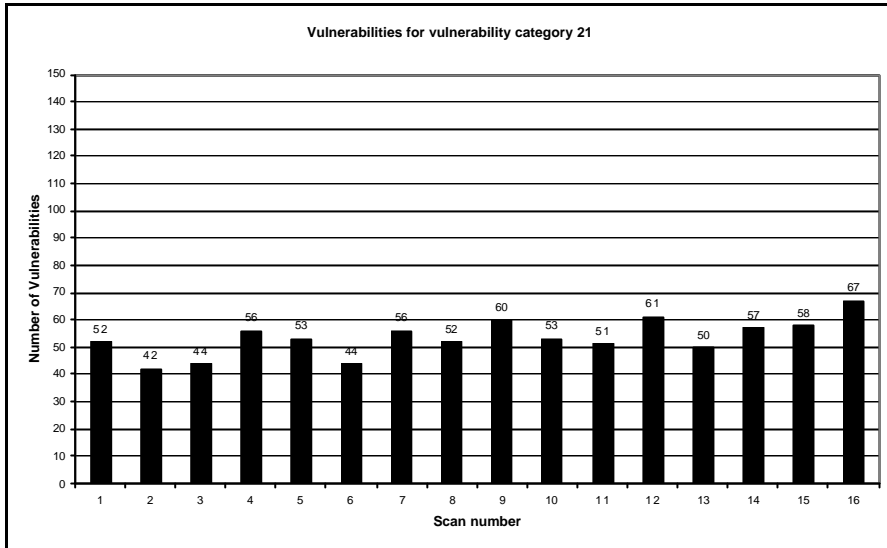


Figure E.37: Scan results over the 16 scans for CyberCop vulnerability category 21

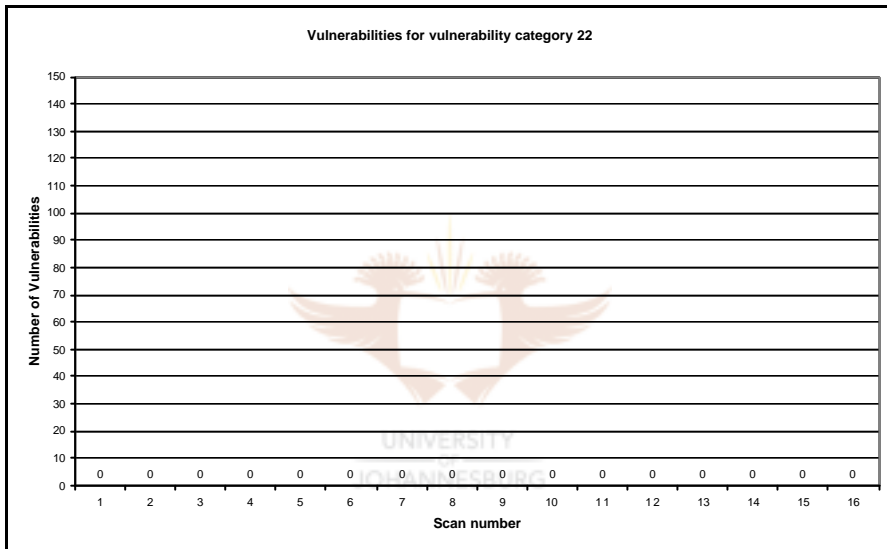


Figure E.38: Scan results over the 16 scans for CyberCop vulnerability category 22

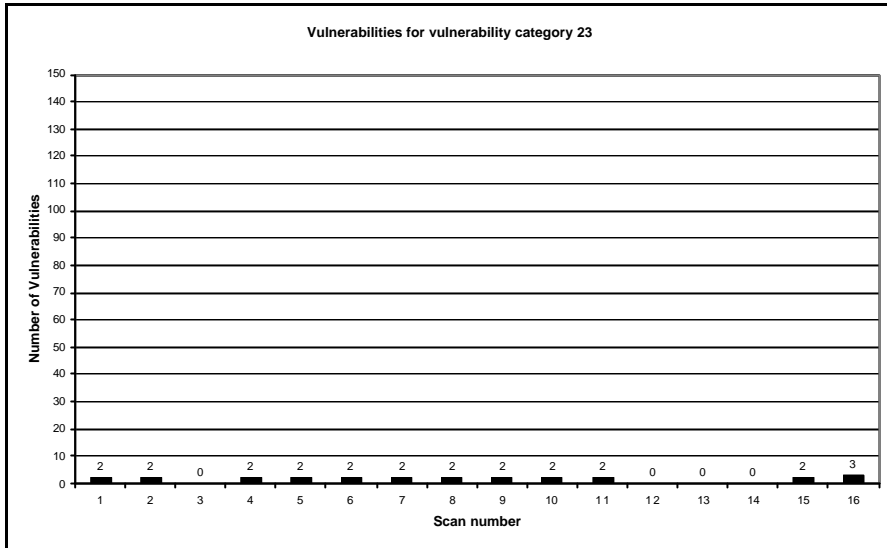


Figure E.39: Scan results over the 16 scans for CyberCop vulnerability category 23

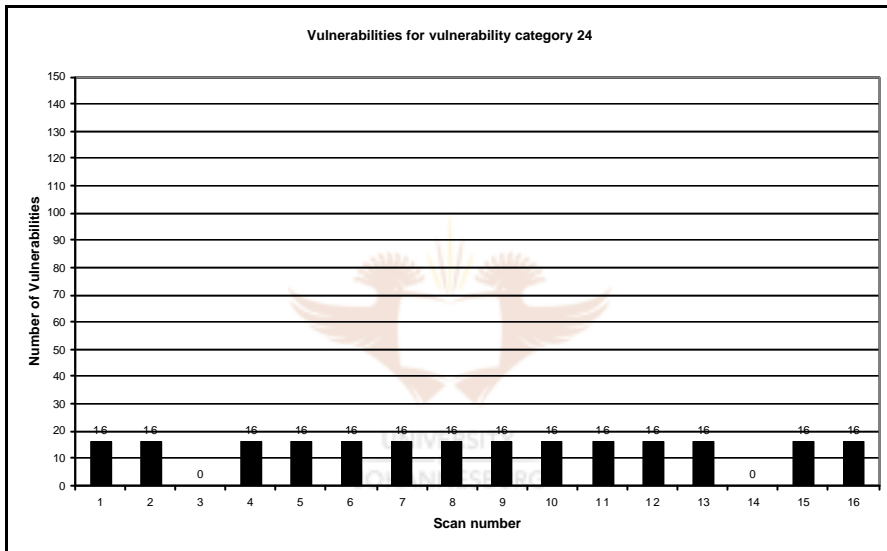


Figure E.40: Scan results over the 16 scans for CyberCop vulnerability category 24

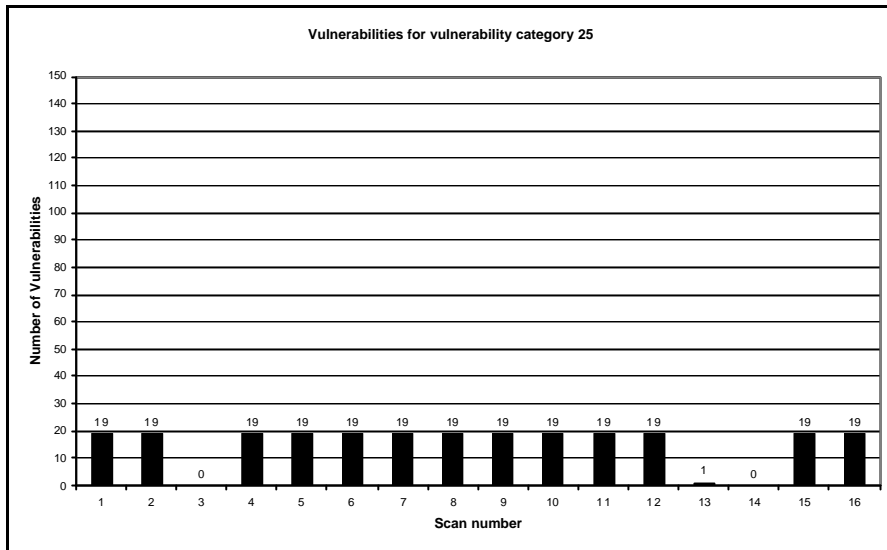


Figure E.41: Scan results over the 16 scans for CyberCop vulnerability category 25

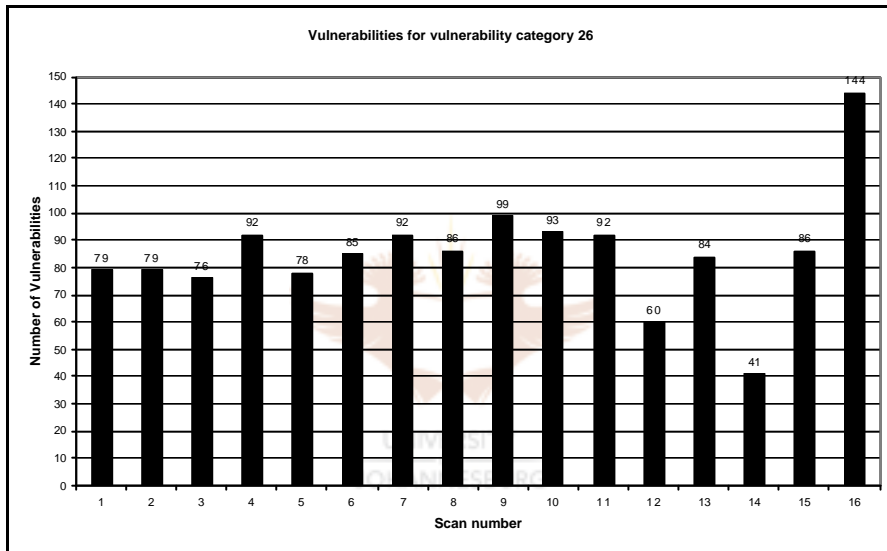


Figure E.42: Scan results over the 16 scans for CyberCop vulnerability category 26

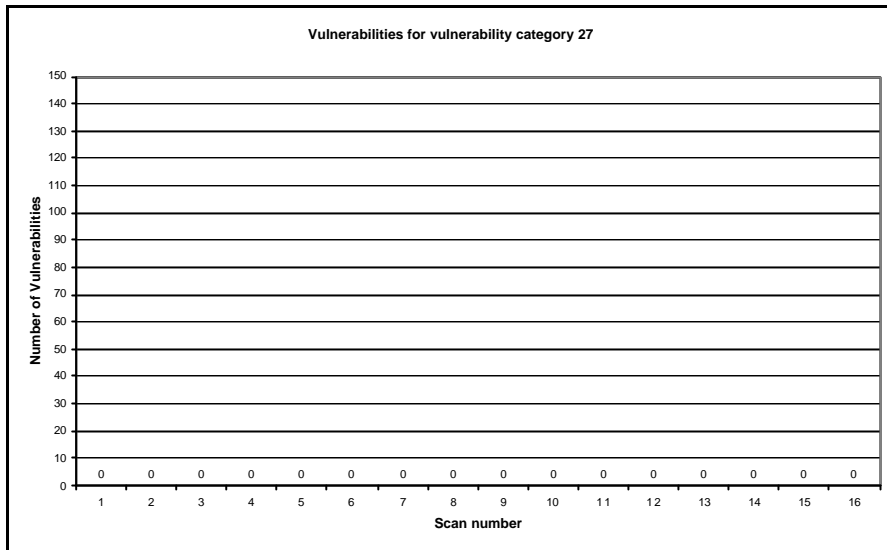


Figure E.43: Scan results over the 16 scans for CyberCop vulnerability category 27

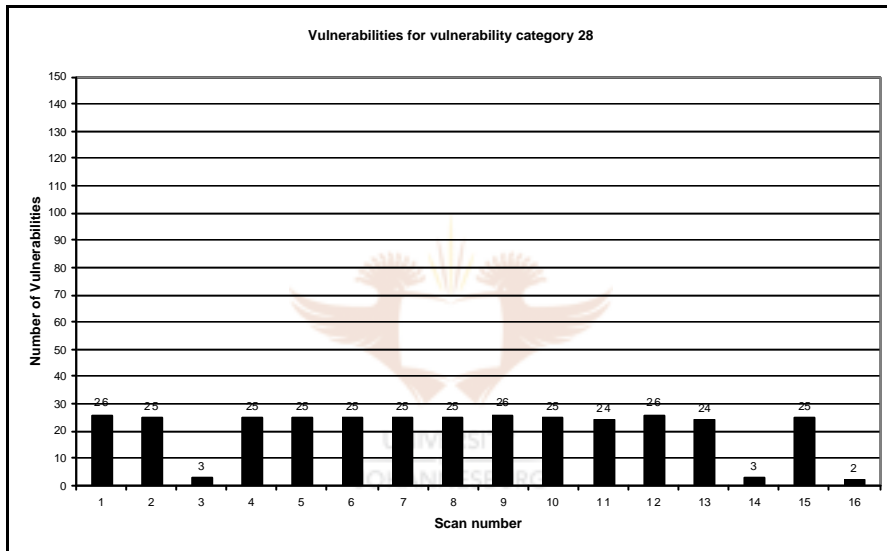


Figure E.44: Scan results over the 16 scans for CyberCop vulnerability category 28

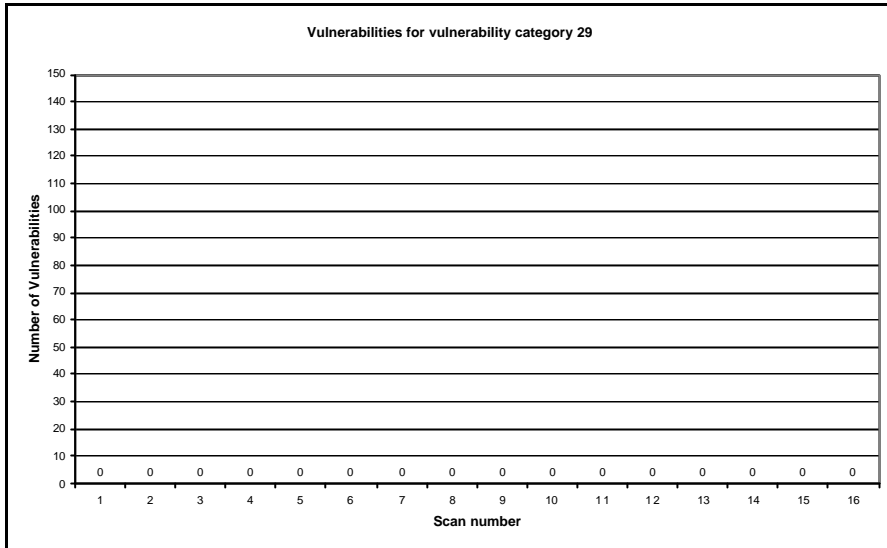


Figure E.45: Scan results over the 16 scans for CyberCop vulnerability category 29

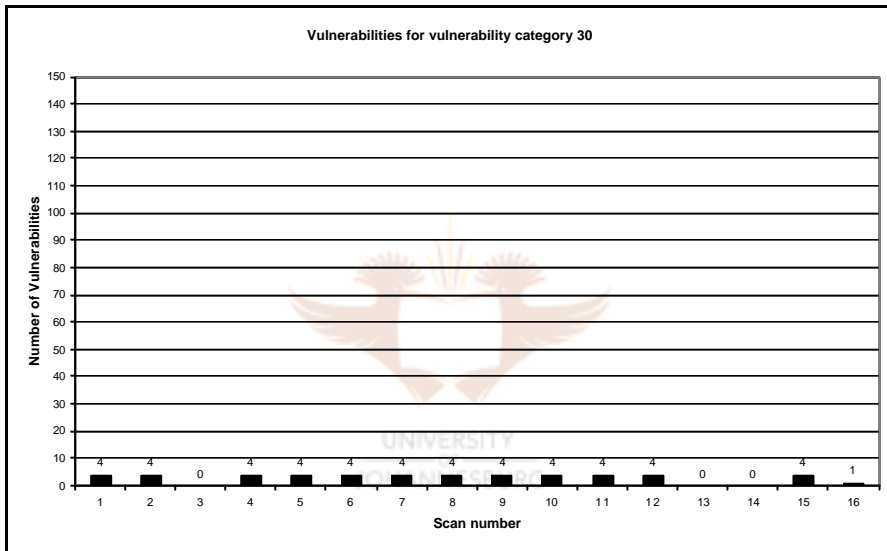


Figure E.46: Scan results over the 16 scans for CyberCop vulnerability category 30

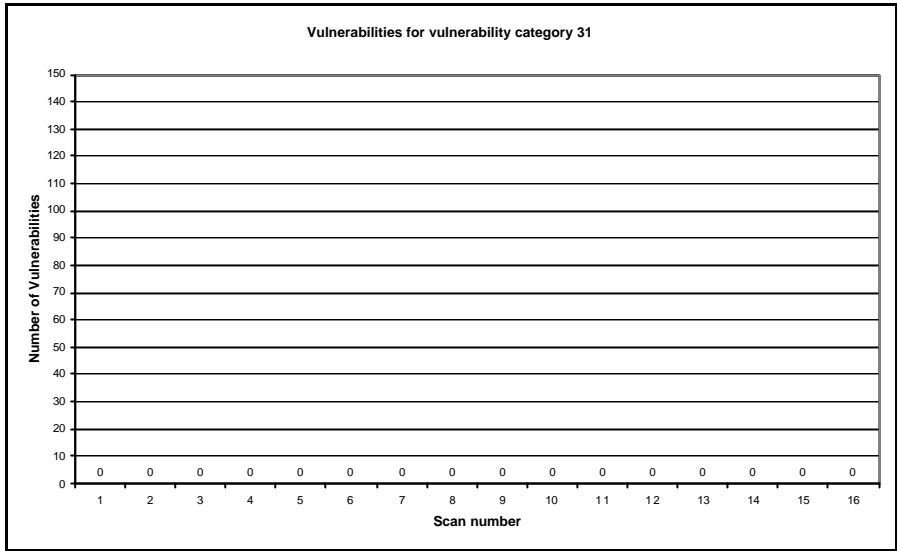


Figure E.47: Scan results over the 16 scans for CyberCop vulnerability category 31





APPENDIX F

PAPERS PUBLISHED

During the course of the research the following papers were prepared and published in a number of journals, while some papers have been submitted for publication, but has not been published yet. In addition, some of the papers have been published in conference proceedings where indicated below.

The following papers have been submitted successfully and were published in journals :

- VENTER, H.S.; ELOFF, J.H.P.; 2000; *Network Security*; “Network Security: Important Issues”; Vol. 6 pp. 12-16; Elsevier Science; ISSN 1353-4858.
- VENTER, H.S.; ELOFF, J.H.P.; 2002; *South African Computer Journal*; “Harmonising Vulnerability Categories”; pp. 24-31; No. 29; Computer Society of South Africa South Africa; ISSN 1015-7999.
- VENTER, H.S.; ELOFF, J.H.P.; 2002; *Computers & Security*; “Vulnerabilities Categories for Intrusion Detection Systems”; Vol. 21 no. 7, pp. 617-619; Elsevier Science; ISSN 0167-4048.
- VENTER, H.S.; ELOFF, J.H.P.; 2003; *Computers & Security*; “A Taxonomy for Information Security Technologies”; Vol. 22; Elsevier Science; ISSN 0167-4048.
- VENTER, H.S.; ELOFF, J.H.P.; 2003; *Network Security*; “Assessment of Vulnerability Scanners”; Vol. 2003 pp. 11-16; Elsevier Science; ISSN 1353-4858.

The following papers have been submitted, but no confirmation has been received up to date:

- VENTER, H.S.; ELOFF, J.H.P.; 2003; *IEEE Intelligent Systems*; “Vulnerability Forecasting”.

- VENTER, H.S.; ELOFF, J.H.P.; 2003; *Computer Communications*, “Vulnerability Forecasting – A Conceptual Model”; Elsevier Science.
- VENTER, H.S.; ELOFF, J.H.P.; 2003; *Computer Networks*, “Vulnerability Scanner Products”; Elsevier Science.
- VENTER, H.S.; ELOFF, J.H.P.; 2003; *Computers & Security*; “State of the Art Intrusion Detection and Vulnerability Scanning”; Elsevier Science; ISSN 0167-4048.

The following papers have been published in conference proceedings:

- VENTER, H.S.; ELOFF, J.H.P.; 2000; IFIP/SEC: *Information Security for Global Information Infrastructures*; Beijing, China; “Network Security Health Checking”; ISBN 7-80003-466-6; pp. 287-220.
- VENTER, H.S.; ELOFF, J.H.P.; 2000; *IT Indaba*, Rand Afrikaans University, South Africa; “A Model for Network Reconfiguration”.
- VENTER, H.S.; ELOFF, J.H.P.; 2001; *Information Security for South Africa (ISSA)*, Magaliesberg Conference Centre, Johannesburg, South Africa; “Dynamic Intrusion Detection Systems”.
- VENTER, H.S.; ELOFF, J.H.P.; 2002; *Information Security for South Africa (ISSA)*, Misty Hills Conference Centre, Johannesburg, South Africa; “Generic Vulnerability Categories”.
- VENTER, H.S.; ELOFF, J.H.P.; 2002; *Annual Conference of the South African Institute of Computer Scientists & Information Technologists (SAICSIT)*, The Boardwalk Conference Centre, Port Elizabeth, South Africa; “Enabling Businesses to Evaluate Intrusion Detection Tools”.

BIBLIOGRAPHY

- [ASTI 99] ASTIHAS, P.; 1999; *Daemon News*; “Intrusion Detection Systems”; <http://www.daemonnews.org/199905/ids.html>.
- [BACE 00] BACE, R. G.; 2000; *Intrusion Detection*; “Defining Intrusion Detection”, pp. 3-4; “Password-Cracking”; pp. 3, 31, 136, 150-151, 179, 279-280; “Intrusion Detection Concepts”, pp. 37-43; “Vulnerability Analysis: A Special Case”, pp. 134-154; “Security as Risk Management”, pp. 258-259; Macmillan Technical Publishing; ISBN 1-57870-185-6.
- [BACK 02] CULT OF A DEAD COW; 2002; *Back Orifice*; <http://www.cultdeadcow.com>.
- [BIND 03] BINDVIEW CORPORATION; 2003; *Proactive security management software and services*; “bv-Control: the security solution to manage within and between organizations”; <http://www.bindview.com>.
- [BISH 99] BISHOP, M.; 1999; *Proceedings of the Recent Advances in Intrusion Detection*; “Vulnerability Analysis”; pp. 125-136.
- [BOBO 95] BOJADZIEV, G.; BOJADZIEV, M.; 1995; *Fuzzy Sets, Fuzzy Logic, Applications*; “Fuzzy Logic”, pp. 177-208; World Scientific Publishing Co Pty. Ltd.; Singapore; ISBN 9-8102-2388-9.
- [BSIB 03] BSI BUSINESS INFORMATION; 2003; *Information Security*; “What is Information Security?”; <http://www.bsi-global.com>.
- [BUGT 02] SECURITYFOCUS.COM; 2002; *Bugtraq*; “Bugtraq Archives”; <http://www.securityfocus.com/forums/bugtraq/intro.html>.
- [CARR 96] CARROLL, J. M.; 1996; *Computer Security*; Butterworth-Heinemann; Third Edition; ISBN 0-7506-9600-1.
- [CEOS 03] CISCO SYSTEMS; 2003; *Product Bulletin, No. 1736*; “End-of-Sale Announcement for Cisco Secure Scanner (Netsonar)”; http://www.cisco.com/warp/public/cc/pd/sqsw/nesn/prodlit/1736_pp.htm.
- [CIDS 03] CISCO SYSTEMS; 2003; *Products & Technologies*; “Cisco Intrusion Detection”; <http://www.cisco.com>.

- [COLE 02] COLE, E.; 2002; *Hackers Beware – Defending Your Network from the Wiley Hacker*; “Install Intrusion Detection Systems”, pp. 238-239; New Riders Publishing; ISBN 0-7357-1009-0.
- [COME 99] COMER, D. E.; 1999; *Computer Networks and Intranets*; “Virtual Private Networks”, p. 191; Prentice Hall; ISBN 0-13-084222-2.
- [COMP 02] 2000 – 2002; *Computers & Security*; Vol. 19 – Vol. 21; Elsevier Science; ISSN 0167-4048.
- [COMP 03] COMPUTER ASSOCIATES; 2003; *Security*; “eTrust Intrusion Detection”; <http://www3.ca.com>.
- [COSL 02] CONWAY, S.; SLIGAR, C.; 2002; *Unlocking Knowledge Assets*; “Building Taxonomies”, pp. 105-124; Microsoft Press; ISBN 0-7356-1463-6.
- [CRMC 01] CRONKHITE, C.; McCULLOUGH, J.; 2001; *Access Denied; “Hackers”*, p. 261; McGraw-Hill/Osborne; ISBN 0-07-213368-6.
- [CSSC 00] CISCO SYSTEMS, INC.; 2000; *Cisco Secure Scanner*; Version 2.0.1.2; <http://www.cisco.com>.
- [CSSC 03] CISCO SYSTEMS; 2003; *Products & Technologies*; “Cisco Secure Scanner”; <http://www.cisco.com>.
- [CYBE 02] NETWORK ASSOCIATES; 2002; *PGP Securities*; “CyberCop Monitor”; <http://www.pgp.com/products/cybercop-monitor/default.asp>.
- [CYBE 03] NETWORK ASSOCIATES; 2003; *Sniffer Technologies*; “CyberCop Scanner”; <http://www.sniffer.com>.
- [DAVI 01] DAVID, J.; 2001; *Network Security*; “The Ins and Outs of Intrusion Detection”; pp. 13-15; Vol. 2001, No. 10; Elsevier Science; ISSN 1353-4858.
- [DENN 87] DENNING, D. E.; 1987; *IEEE Transactions on Software Engineering*; “An Intrusion-Detection Model”; pp. 222-232; Vol. 13, No. 2; IEEE Computer Society; ISSN 0098-5589.
- [DERA 03] DERAISON, R.; 2003; *Nessus*; “What is Nessus?”; <http://www.nessus.org/intro.html>.
- [EEYE 03] EEYE DIGITAL SECURITY; 2003; *Retina Network Security Scanner*; “Superior Vulnerability Detection & Remediation”; <http://www.eeye.com/html/Products/Retina/index.html>.

- [ESCI 02] ENTERPRISE SYSTEMS CONSULTING INCORPORATED; 2002; *Intrusion Detection & Vulnerability Assessment*; “Managing Risk through Technology”; <http://www.lesc.com>.
- [FRAU 02] 2000 – 2002; *Computer Fraud & Security*; Vol. 2000 – Vol. 2002; Elsevier Science; ISSN 1361-3723.
- [GENG 02] GENGLER, B.; 2001; *Computer Fraud & Security*; “Intrusion Detection Systems New to Market”; p. 4; Vol. 2002, No. 5; Elsevier Science; ISSN 1361-3723.
- [GOLL 99] GOLLMANN, D.; 1999; *Computer Security*; “Computer Security”, p. 5; “Layered models”, pp. 225-226; John Wiley & Sons; ISBN 0-471-97844-2.
- [GRAH 00] GRAHAM, R.; 2000; *RobertGraham.com*; “FAQ: Network Intrusion Detection Systems”; <http://www.robertgraham.com/pubs/network-intrusion-detection.html>.
- [GREE 02] GREENSTEIN, M.; VASARHELYI, M.; 2002; *Electronic Commerce – Security, Risk Management, and Control*; Second Edition; “Risks Associated with Viruses and Malicious Code Overflows”, pp. 242-245; McGraw-Hill; ISBN 0-07-241081-7.
- [HAMB 93] HARRIS, C. J.; MOORE, C. G.; BROWN, M.; 1993; *Intelligent Control – Aspects of Fuzzy Logic and Neural Nets*; “Neural Network Approximation Capability for Control and Modelling”, pp. 282-312; “The B-Spline Neural Network and Fuzzy Logic”, pp. 314-356; World Scientific; ISBN 981-02-1042-6.
- [HANC 01] HANCOCK, B.; 2001; *Security Views*; “U.S. DoD Puts Up Blocks to Code Red”, pp. 451-452; Elsevier Science; ISSN 0167-4048.
- [HARR 03] HARRIS; 2003; *STAT – Security Threat Advance Technology*; “STAT Scanner Professional”; <http://www.statonline.com>.
- [HILL 02] HILLEY, S.; 2002; *Computer Fraud & Security*; “Where to buy stolen credit cards!”; p. 2; Vol. 2002, No. 6; Elsevier Science; ISSN 1361-3723.
- [HUTH 01] HUTH, M. R. A.; 2001; *Secure Communicating Systems – Design, Analysis, and Implementation*; Cambridge University Press; ISBN 0-521-80731-X.

- [IFAC 98] INTERNATIONAL FEDERATION OF ACCOUNTANTS; 1998; *International Information Technology Guideline*; “Managing Security of Information”; ISBN 1-887-464-31-X.
- [INFO 02] INFOSEC; 2002; *Information Security & Prevention of Computer Related Crime*; “What is Information Security?”; http://www.infosec.gov.hk/english/general/infosec/what_infosec.htm.
- [INSE 03] INSECURE.ORG; 2003; *Introduction*; “Nmap Stealth Port Scanner”; <http://www.insecure.org>.
- [IPFA 03] IPFAQ; 2003; *IPv6 Information Page*; “IP FAQ”; <http://www.ipv6.org>.
- [IPV6 03] IPV6 FORUM; 2003; *IPv6 Forum*; “About the IPv6 Forum”; <http://www.ipv6forum.com>.
- [IPVE 03] SUN MICROSYSTEMS; 2003; *IP Version 6*; “Introduction”; <http://playground.sun.com/pub/ipng/html/ipng-main.html>.
- [ISOR 89] INTERNARIONAL STANDARDS ORGANIZATION; 1989; *ISO 7498-2: Information Processing Systems – Open System Interconnection*; “Basic Reference Model – Part 2: Security Architecture”; <http://www.iso.ch>.
- [ISSC 03] INTERNET SECURITY SYSTEMS; 2003; *Vulnerability Assessment*; “Internet Scanner®”; <http://www.iss.net>.
- [ISSN 03] INTERNET SECURITY SYSTEMS; 2003; *Internet Security Systems*; “ISS”; <http://www.iss.net>.
- [JAHN 02] JAHNKE, M.; 2002; *Research Establishment for Applied Science*; “SIDI - An Implementation of a Survivable Intrusion Detection Infrastructure”; <http://www.fgan.de/~jahnke/sidi>.
- [KABY 78] KANDEL, A.; BYATT, W.J.; 1978; *Proceedings of the IEEE*; “Fuzzy Sets, Fuzzy Algebra and Fuzzy Statistics”; Vol. 66; No. 12; pp. 1619-1631.
- [KAND 82] KANDEL, A.; 1992; *Fuzzy Techniques in Pattern Recognition*; “Fuzzy Sets”, pp. 23-43; John Wiley & Sons Inc.; ISBN 0-471-09136-7.
- [KAND 92] KANDEL, A.; 1992; *Fuzzy Expert Systems*; “General Purpose Fuzzy Expert Systems”, pp. 23-41; CRC Press Inc.; ISBN 0-8493-4297-X.

- [KASA 00] KANLAYASIRI, U.; SANGUANPONG, S.; 2000; *Proceedings of the 7th International Workshop on Academic Information Networks and Systems, Bangkok, Thailand*; “Network-based Intrusion Detection Model for Detecting TCP SYN Flooding”.
- [KEOS 01] KING, C. M.; DALTON, C. E.; OSMANOGLU, T. E.; 2001; *Security Architecture – Design, Deployment & Operations*; “Security Policies, Standards, and Guidelines”, pp. 13-39; RSA Press/Osborne/McGraw-Hill; ISBN 0-07-213385-6.
- [KIDO 01] KING, C. M.; DALTON, C. E.; OSMANOGLU, T. E.; 2001; *Security Architecture – Design, Development & Operations*; “Business and Application Drivers (Case Study)”, p. 1; “Authorisation and Access Control”, pp. 93-94; “Basic Intrusion Detection Terminology”, pp. 287-288; McGraw-Hill/Osborne; ISBN 0-07-213385-6.
- [KUSP 95] KUMAR, S.; SPAFFORD, E. H.; 1995; *Computers & Security*; “A Software Architecture to Support Misuse Intrusion Detection”, p. 607; Vol. 14, No. 7; Elsevier Science; ISSN 0167-4048.
- [LEXI 02] LEXICO LLC; 2002; *Dictionary.com*; “technology”; “password”; <http://www.dictionary.com>.
- [LEXI 03] LEXICO LLC; 2003; *Dictionary.com*; “architecture”; “risk”; “scan”; “vulnerability”; <http://www.dictionary.com>.
- [LITS 01] LIN, Y. T.; TSENG, S. S.; LIN, S. C.; 2001; *Journal of Information Science and Engineering*; “An Intrusion Detection Model Based Upon Intrusion Detection Markup Language”, pp. 899-919; Vol. 17, No. 6; ISSN 1016-2364.
- [LOPH 02] @STEAK.COM; 2002; *L0pht Crack*; “L0pht Crack Version 3.0”; <http://www.atstake.com/research/lc3/index.html>.
- [LOPY 01] LOPYREV, A.; 2001; *SANS Info Sec Reading Room*; “Distributed Scan Model for Enterprise-Wide Network Vulnerability Assessment”; http://www.sans.org/rr/audit/scan_model.php.
- [MAIW 03] MAIWALD, E.; 2003; *Network Security: A Beginner's Guide*; Second Edition; “Defining Information Security”, p. 4; Osborne McGraw-Hill; ISBN 0-07-2229-578.

- [MASI 02] MAIWALD, E.; SIEGLEIN, W.; 2002; *Security Planning & Disaster Recovery*; “Information Security Policy”, p. 61; McGraw-Hill/Osborne; ISBN 0-07-222463-0.
- [MCAF 03] MCAFEE SECURITY; 2003; *CyberCop ASaP*; “Vulnerability Assessment”; <http://www.mcafeeb2b.com/services/cybercop-asap.asp>.
- [MCLE 00] McLEAN, I.; 2000; *Windows 2000 Security – Little Black Book*; The Coriolis Group; ISBN 1-57610-387-0.
- [MCSK 02] McCLURE, S.; SCAMBRAY, J.; KURTZ, G.; 2002; *Hacking Exposed*; Third Edition; “Denial of Service (DoS) Attacks”, pp. 503-525; “Cryptography”, p. 581; McGraw-Hill/Osborne; ISBN 0-07-219382-4.
- [MICR 03] MICROSOFT; 2003; *Microsoft COM Technologies*; “ActiveX Controls”; <http://www.microsoft.com/com/tech/ActiveX.asp>.
- [MITR 03] THE MITRE CORPORATION; 2003; *Common Vulnerabilities and Exposures (CVE)*; “CVE, The Key to Information Sharing”; <http://www.cve.mitre.org/introduction.html>.
- [MOHA 01] MOHAN, P.; 2001; *SANS Info Sec Reading Room*; “Need for Pure Integration between Intrusion Detection and Vulnerability Assessment”; <http://www.sans.org/rr/intrusion/integration.php>.
- [NETB 02] SPECTROSOFT; 2002; *Netbus*; <http://www.netbus.org>.
- [NETI 03] NETIQ; 2003; *Products and Solutions*; “Security Analyzer”; <http://www.netiq.com>.
- [NETR 02] SYMANTEC; 2002; *Products*; “Symantec NetRecon 3.5”; <http://enterprisesecurity.symantec.com/products/products>.
- [NETW 02] 2000 – 2002; *Network Security*; Vol. 2000 – Vol. 2002; Elsevier Science; ISSN 1353-4858.
- [NETW 03] NETWORK ASSOCIATES; 2003; *CyberCop Scanner*; “CyberCop Scanner ASaP”; <http://www.networkassociates.com>.
- [NFRS 03] NFR SECURITY; 2003; *NFR Intrusion Management System*; “NFR Network Intrusion Detection”; <http://www.nfr.com/products/NID>.
- [NOCF 01] NORTHCUTT, S.; COOPER, M.; FEARNOW, M.; FREDERICK, K.; 2001; *Intrusion Signatures and Analysis*; “Passwords”,

- pp. 57-65, 76-85, 134-143, 149-168, 189, 233-250. New Riders Publishing; ISBN 0-7357-1063-5.
- [NONM 01] NORTH CUTT, S.; NOVAK, J.; McLACHLAN, D.; 2001; *Network Intrusion Detection – An Analyst’s Handbook*; Second Edition; New Riders Publishing; ISBN 0-7357-1008-2.
- [NORT 01] NORTH CUTT, S.; NOVAK, J.; McLACHLAN, D.; 2001; *Network Intrusion Detection*; Second Edition; “Misconfigured Systems”, pp. 159, 188, 213-214, 387-391; New Riders Publishing; ISBN 0 7357-1008-2.
- [OPPL 98] OPPLIGER, R.; 1998; *Internet & Intranet Security*; “Access Control Mechanisms”, p. 58; “Access Control”, pp. 91-147; Artech House Incorporated; ISBN 0-89006-829-1.
- [PAGU 96] PABRAI, U. O.; GURBANI, V. K.; 1996; *Internet & TCP/IP Network Security – Securing Protocols and Applications*; “Firewall Systems”, pp. 163-181; McGraw-Hill; ISBN 0-07-048215-2.
- [PALM 01] PALMER, C.; 2001; *Network Security*; “Nimda virus hits”; p. 4; Vol. 2001, No. 9; Elsevier Science; ISSN 1353-4858.
- [PGPI 03] ZIMMERMANN, P.; 2003; *PGP International Home Page*; “PGP”; <http://www.pgpi.org>
- [PHLE 03] PHLEEGER, C. P.; 2003; *Security in Computing*; Third Edition; “Encryption Algorithms”, pp. 37-39; “The Caesar Cipher”, pp. 41-44; “Hash Algorithms”, pp. 76-77; “Certificates”, pp. 81-82; “Mandatory and Discretionary Access Control”, p. 256; Prentice Hall; ISBN 0-13-035548-8.
- [REAL 02] INTERNET SECURITY SYSTEMS; 2002; *Internet Security Systems Incorporated*; “RealSecure Gigabit Network Sensor 7.0”; <http://www.iss.net>.
- [REAL 03] INTERNET SECURITY SYSTEMS; 2003; *Internet Security Systems Incorporated*; “RealSecure Gigabit Network Sensor 7.0”; <http://www.iss.net>.
- [ROSH 03] ROSHAL, E.; 2003; *RARLAB*; “WinRAR”; <http://www.rarlab.com>.
- [SAIN 03] SAINT CORPORATION; 2003; *About SAINT*; “SAINT 4 Vulnerability Assessment Tool”; <http://www.saintcorporation.com>.

- [SCHN 00] SCHNEIER, B.; 2000; *Secrets and Lies – Digital Security in a Networked World*; “Hackers”, pp. 43-46; “Intrusion Detection Systems”, pp. 194-197; John Wiley & Sons Inc.; ISBN 0-471-25311-1.
- [SCHN 88] SCHNEIDER, M.; 1988; *Fuzzy Sets and Systems*; “Properties of the Fuzzy Expected Value and the Fuzzy Expected Interval in Fuzzy Environment”; pp. 55-68; Vol. 28; Elsevier Science Publishers; ISSN 0165-0114.
- [SCHU 03] SCHULTZ, E.; 2003; *Computers & Security*; “Internet security: what’s in the future?”, pp. 78-79; Vol. 22, No. 2; Elsevier Science; ISSN 0167-4048.
- [SCMK 01] SCAMBRAY, J.; McCLURE, S.; KURTZ, G.; 2001; *Hacking Exposed*; Second Edition; “Footprinting”, pp. 5-34, 87-95, 164-174, 238-241, 252-257, 287-290, 308, 339-340, 433-437, 453-456, 483-506, 507-653; Osborne/McGraw-Hill; ISBN 0-07-212748-1.
- [SECF 02] SECURITY FOCUS; 2002; *Advisories*; “IIS Worms Detector”; <http://www.securityfocus.com>.
- [SMIT 00] SMITH, E.; 2000; *Information Security In Health-Care Systems: A New Approach To It Risk Management*; “Basic concepts behind fuzzy system modeling”; p. A-2.; RAU Library, Rand Afrikaans University, Kingsway, Auckland Park, Johannesburg, South Africa.
- [SNOR 02] SNORT.ORG; 2002; *Snort*; “Snort”; “The Open Source Network Intrusion Detection System”; <http://www.snort.org>.
- [STEI 98] STEIN, L. D.; 1998; *Web Security – A Step-by-Step Reference Guide*; “Certifying Authorities and the Public Key Infrastructure”, pp. 25-28; Addison Wesley; ISBN 0-201-63489-9.
- [SYMA 03] SYMANTEC ENTERPRISE SOLUTIONS; 2003; *Network vulnerability assessment with root cause analysis*; “Symantec NetRecon 3.5”; <http://enterprisesecurity.symantec.com>.
- [TALI 00] TALISKER; 2000; *Network Vulnerability Scanners*; “Nessus”; http://www.networkintrusion.co.uk/N_scan.htm.
- [TIWA 99] TIWANA, A.; 1999; *Web Security*; “Are Firewalls Enough?”, pp. 112-135; “Securing Transactions with Digital Certificates”, pp. 211-227; Digital Press; ISBN 1-55558-210-9.

- [TRIU 02] TRIULZI, A.; 2002; *K2 Defender Intrusion Detection System*; “Rising the Challenge: Moving from Intrusion Detection to Security Monitoring”; <http://www.k2defender.com>.
- [TUDO 00] TUDOR, J. K.; 2000; *Information Security Architecture – An Integrated Approach to Security in the Organization*; Auerbach; ISBN 0-8493-9988-2.
- [VEE1 03] VENTER, H. S.; ELOFF; J. H. P; 2003; *Computers & Security*; “A Taxonomy for Information Security Technologies”; Elsevier Science; ISSN 0167-4048.
- [VEE2 03] VENTER, H. S.; ELOFF; I H. P; 2003; *South African Computer Journal*; “Harmonised Vulnerability Categories”; pp. 24-31; No. 29; Computer Society of South Africa South Africa; ISSN 1015-7999.
- [VENT 03] VENTER, H. S.; 2003; *Department of Computer Science, University of Pretoria, Pretoria, South Africa*; “The VF Prototype”; hventer@cs.up.ac.za.
- [VISS 03] VISSER, P.; 2003; *PO Box 746, Florida Hills, 1716, South Africa*; “The VF Prototype”; visser_pierre@hotmail.com.
- [WACA 98] WALKER, K. M.; CAVANAUGH, L. C.; 1998; *Computer Security Policies and SunScreen Firewalls*; Prentice Hall; ISBN 0-13-096015-2.
- [WEBA 03] WEBOPEDIA.; 2003; *Terms*; “AES”; <http://www.webopedia.com/term/a/aes.html>.
- [WEBD 03] WEBOPEDIA.; 2003; *Terms*; “DES”; <http://www.webopedia.com/term/d/des.html>.
- [WEBP 03] WEBOPEDIA.; 2003; *Terms*; “Public-key cryptography”; http://www.webopedia.com/term/p/public_key_cryptography.html.
- [WEBR 03] WEBROOT.; 2003; *Products*; “Privacy Master”; <http://www.webroot.com/wb/products/privacymaster/index.php>.
- [WEBS 03] WEBOPEDIA.; 2003; *Terms*; “Symmetric-key cryptography”; http://www.webopedia.com/term/s/symmetric_key_cryptography.htm.
- [YAZA 92] YAGER, R. R.; ZADEH, L. A.; 1992; *An Introduction to Fuzzy Logic Applications in Intelligent Systems*; Kluwer Academic Publishers; ISBN 0-7923-9191-8.

- [YELZ 95] YEN, J.; LANGARI, R.; ZADEH, L. A.; 1993; *Industrial Applications of Fuzzy Logic and Intelligent Systems*; “Two-Degree-of-Freedom Fuzzy Model for Flight”, p. 119; IEEE Press; ISBN 0-7803-1048-9.
- [ZADE 65] ZADEH, L. A.; 1965; *Information Control*; “Fuzzy Sets”; Vol. 8; p. 338.

