A NEW MODEL FOR WORM DETECTION AND RESPONSE

Madihah MOHD SAUDI

PhD

UNIVERSITY OF BRADFORD

2011

A NEW MODEL FOR WORM DETECTION AND RESPONSE

Development and evaluation of a new model based on knowledge discovery and data mining techniques to detect and respond to worm infection by integrating incident response, security metrics and apoptosis.

Madihah MOHD SAUDI

Submitted for the degree of

Doctor of Philosophy

Department of Computing

School of Computing, Informatics and Media

University of Bradford

2011

A NEW MODEL FOR WORM DETECTION AND RESPONSE

Madihah MOHD SAUDI

ABSTRACT

KEYWORDS:
Apoptosis, data mining, security metrics, Knowledge Discovery Technique (KDD), Standard Operating Procedures (SOP), worm incident response, static analysis, dynamic analysis, worm rules and worm classification.

Worms have been improved and a range of sophisticated techniques have been integrated, which make the detection and response processes much harder and longer than in the past. Therefore, in this thesis, a STAKCERT (Starter Kit for Computer Emergency Response Team) model is built to detect worms attack in order to respond to worms more efficiently.

The novelty and the strengths of the STAKCERT model lies in the method implemented which consists of STAKCERT KDD processes and the development of STAKCERT worm classification, STAKCERT relational model and STAKCERT worm apoptosis algorithm. The new concept introduced in this model which is named apoptosis, is borrowed from the human immunology system has been mapped in terms of a security perspective. Furthermore, the encouraging results achieved by this research are validated by applying the security metrics for assigning the weight and severity values to trigger the apoptosis. In order to optimise the performance result, the standard operating procedures (SOP) for worm incident response which involve static and dynamic analyses, the knowledge discovery techniques (KDD) in modeling the STAKCERT model and the data mining algorithms were used.

This STAKCERT model has produced encouraging results and outperformed comparative existing work for worm detection. It produces an overall accuracy rate of 98.75% with 0.2% for false positive rate and 1.45% is false negative rate. Worm response has resulted in an accuracy rate of 98.08% which later can be used by other researchers as a comparison with their works in future.

ACKNOWLEDGMENTS

I would like to express my deepest thanks and gratitude to Dr Andrea J Cullen and Professor M. E. Woodward for their advice, professional guidance, attention and support throughout this research. This research could not have been possible without their guidance and brilliant ideas.

Thanks to all the research students I have met through these years for ideas and inspiration; with special thanks to Dr. Solahuddin, each and every one of you have contributed in your own special way.

Most of all, I am grateful to my husband, Masrur Ibrahim, for his tireless encouragement and patience. I also thank my children, Harith Muanis and Nuha Najiyyah for cheering me up at a time when I needed it most. Finally, I wish to extend my heartfelt appreciation to my family and relatives for their unconditional love and support.

This research is fully sponsored by the Ministry of Higher Education, Malaysia and Universiti Sains Islam Malaysia (USIM).

PUBLICATIONS

- Saudi,M. M., M.Tamil, E., Cullen,A.J., Woodward, M., I.Idris,M.Y.(April 2009) Reverse Engineering: EDOWA Worm Analysis and Classification. In: Ao,S.I.& Gelman,L.,eds. *Advances in Electrical Engineering and Computational Science*, *Lecture Notes in Electrical Engineering.* Berlin: Springer Netherlands, pp. 277-288.
- Saudi,M.M., Cullen,A.,Woodward,M. (2009) An overview of STAKCERT Framework in Confronting Worms Attack, *2nd IEEE International Conference on Computer Science and Information Technology (IEEE ICCSIT 2009)*, *Conference Proceedings,* v3*,* Beijing, China, 8 – 11th August 2009.IEEE Xplore: IEEE Press, pp.104-108.
- Saudi,M.M., Cullen,A.,Woodward,M. (2009) STAKCERT Framework in Eradicating Worms Attack. *2009 International Conference on CyberWorlds*, *Conference Proceedings*, University of Bradford, UK, 7-11th September 2009, IEEE Computer Society, pp. 257-264.
- Saudi,M.M, Cullen, A.J. and Woodward, M.E. (2010) STAKCERT Worm Relational Model for Worm Detection, Lecture Notes in Engineering and Computer Science: Proceedings of The World Congress on Engineering 2010, WCE 2010, 30 June-2 July, 2010, London, U.K.pp 469-473.
- Saudi,M.M, Cullen, A.J. and Woodward, M.E. (2010) Statistical Analysis in Evaluating STAKCERT Infection, Activation and Payload Methods, Lecture Notes in Engineering and Computer Science: Proceedings of The World

Congress on Engineering 2010, WCE 2010, 30 June-2 July, 2010, London, U.K.pp 474-479. *(received award from organizer IAENG committees: Certificate of Merit –Student)*

- Saudi,M.M, Cullen, A.J. and Woodward, M.E. (2011) Efficient STAKCERT KDD Processes in Worm Detection, Proceedings of the International Conference of Computer and Information Science 2011, 27 July-29 July 2011,Paris, France, pp 428-432.
- Saudi,M.M., Cullen, A.J. and Woodward, M. (2011), Efficient STAKCERT KDD Processes in Worm Detection, World Academy of Science, Engineering and Technology Journal, Issue 79, pp. 453-457 (pISSN 2010-376X, eISSN 2010-3778).

TABLE OF CONTENTS

CHAPTER 5 MODELLING STAKCERT FOR WORM RESPONSE

LIST OF FIGURES

List of Tables

List of Abbreviations

CBR	Case-based Reasoning
CERT	Computer Emergency Response Team
DoS	Denial of Service
DDoS	Distributed Denial of Service
ED	Euclidean Distance
FN	False Negative
FNR	False Negative Rate
FP	False Positive
FPR	False Positive Rate
IBk	Represents K-nearest neighbor classifier
IDS	Intrusion Detection System
IR	Incident Response
J48	Represents one of the classifiers in Decision Tree
KDD	Knowledge Discovery Database
LAN	Local Area Network
MLP	Multilayer Perceptron
MyCERT	Malaysia Computer Emergency and Response Team
ROC	Receiver-Operator Characteristics
SMO	Sequential Minimal Optimisation
SOP	Standard Operating Procedures
STAKCERT	Starter Kit for Computer Emergency Response Team
TP	True Positive
TPR	True Positive Rate
TN	True Negative
TNR	True Negative Rate
WEKA	Waikato Environment for Knowledge Analysis

CHAPTER 1

INTRODUCTION

This chapter gives a brief background to the current state of the cyber threats posed by worms. Moreover the research motivation, and the research aims and objectives are laid out in order to give the reader a glimpse of what inspired this research. The original contributions and the thesis structure are also covered in this chapter.

1.1 Background

Computer worms have become a real threat to computer users for more than a decade. Worms reproduce themselves and defensive measures have focused on stopping or slowing their spread. Ultimately, though, there is no defence better than a comprehensive security strategy that embraces user education, crisis-response teams and technologically sound security measures including, but not limited to, those that relate specifically to the threats posed by worms (Saudi and Jomhari 2006, Hawkins *et al.* 2000). Defence against harm can consist of preventing the harm from occurring, limiting the extent of the harm or recovering from the harm after it has occurred. For the past few years, incident response has been seen as one of the fastest growing and most important issues in computer security (Killcrece *et al*. 2003, Schultz 2007). An incident is referred to as an adverse event that threatens security in computing systems and networks (Schultz 2007). When an incident occurs, the action which can be triggered by human or computer systems is known as incident response. The incident response not only helps to minimise the damage from the incident, but also takes possible incidents into consideration and prepares the system before the incident happens (Schultz and Shumway 2001). Nowadays, it is really hard to think about confronting a cyber world incident without integrating the approach with an incident response procedure. Incident response helps to minimise the damage caused by security incident by providing the standard operating procedures on how to react and solve for each security incident. Furthermore, Schultz (2007) states that the potential growth of computer forensics in future, which is part of the incident response itself. With improvements in technology, vulnerabilities in systems or applications can be easily exploited in a fraction of a second. Statistics taken from CyberSecurity (2010) show that three types of major security incidents are often reported (i.e. fraud, intrusion and malicious code) as displayed in Figure 1.1

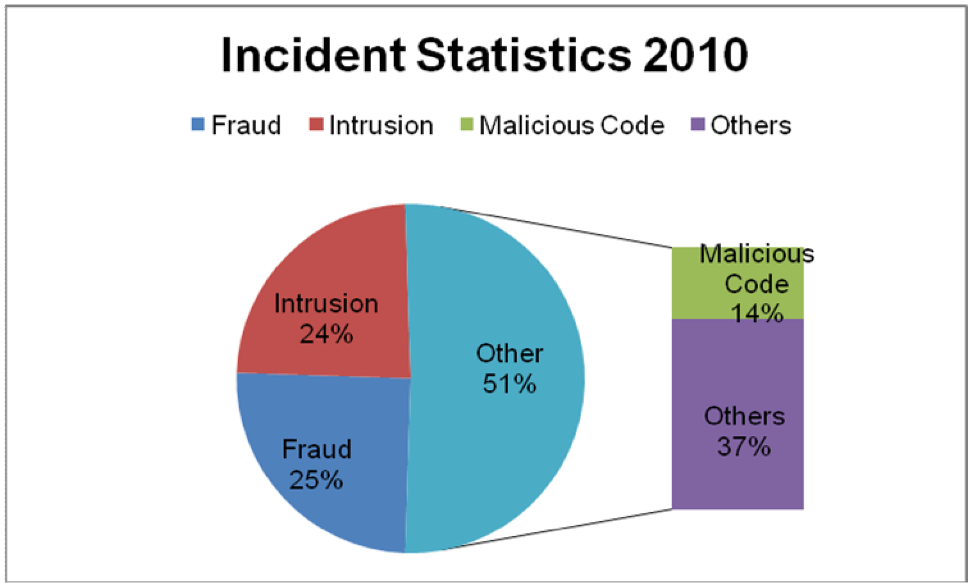


Figure 1.1. Incident Statistics for 2010 in Malaysia.
(Adapted from CyberSecurity Malaysia Incident Statistics (2010))

Indeed, based on a report by the New Zealand Computer Crime and Security Survey 2010 (Quinn 2010) it would appear that the two most costly and most widely experienced computer security incidents were related to virus contamination and malware infection, each of which contributed 37% and 22% respectively. Furthermore, in this survey it is stated that many organisations were struggling to detect and remove the Conficker worm and millions of computers around the world were infected by it. Also in this survey, it stated that fake anti-virus software installed on a victim's computer is another common problem faced by many users and organisations. It took several days of effort to detect and remove this malicious software. Therefore, if the incident response is being applied in an organization, there is possibility that these issues can be solved faster and efficiently. Furthermore, if a comparison between current trends and those of 10 years ago is made, these historical worm attacks and infections ensured the reputation of the attacker and thus gaining respect from other attackers or hackers was paramount (Whitty 2007). In contrast, the motivation for cyber attacks in the past 5 years is profit. Currently attackers try to steal passwords or credit card information by phishing, installed keyloggers and backdoors on end user computers to steal confidential information or are involved with launching bots from end user computers to perform ongoing attacks. These cyber attacks have caused loss of millions which is estimated around $1 billion and productivity for many organisations and end users (Liu and Uppala 2006).

There are numerous ways to handle worm incidents. These include keeping anti-virus software updated (MyCERT 2009a), keeping the operating system updated with the latest patches (Microsoft 2008), not opening unknown email attachments and never following links that ask for id and passwords for online banking purpose (MyCERT 2009b). Unfortunately, there are still many users who lack the experience or the knowledge to detect when their computers are infected by worms (Schroeder 2005).

In this research, a model called the STAKCERT model is introduced. The acronym 'STAKCERT' stands for Starter Kit for Computer Emergency Response Team. It is a new model which consists of STAKCERT worm classification, STAKCERT relational model, STAKCERT worm apoptosis algorithm and with an improvement method which is called as STAKCERT KDD Processes (refer Figure 3.5).This STAKCERT model is an improvement on the traditional incident detection and response models. This STAKCERT model is built to detect and respond to worm incidents. It is the integration of incident response, data mining, security metrics and the apoptosis concept. The STAKCERT model has been simulated and tested, and it is indicated that this model has successfully detected and responded to worms with a detection accuracy rate of 98.75% and a response accuracy rate of 98.08%. This detection accuracy result outperformed the comparative existing works (refer to Chapter 4, section 4.4.3). As for worm response, this thesis has provided an accuracy rate which in future can be used by other researchers as a comparison with their works (refer Chapter 5,section 5.4). Moreover, in terms of worm response, the STAKCERT model has introduced a new concept called apoptosis, which is useful for indicating if the victim's computer is severity affected by worm infection. Apoptosis is also known as cell-programmed death is a concept borrowed from human immunology system. In this thesis, the apoptosis has been mapped into worm's perspective, where the network connection of a severely infected computer will be disconnected to avoid the further propagation of the worm Based on the results achieved and the contributions made, this research has successfully achieved all the objectives it targeted.

1.2 Motivation

The motivation for this research came from industries' needs for a reliable worm detection and response model in order to defend information infrastructure (Liu and Uppala 2006, Nicol 2005). The researcher's experiences with the real world during an attachment with the Malaysian Computer Emergency Response Centre (MyCERT) has given this research much needed input and insight into the problems faced by industry and by the public at large. A lack of understanding and knowledge and proper procedures for worm detection and response have led to money loss, reduced productivity and the tarnishing of organisation's reputations (Mitropoulos *et al.* 2006).

Currently the worm characteristics have been greatly improved and more sophisticated techniques have been integrated, which make the detection and removal processes harder, and which take longer than has been the case in the past (Smith 2008). From 2001 to 2011, worm attacks mainly focused on exploiting vulnerabilities in applications and in operating system such as in Windows (Schneier 2005, Microsoft 2011, Kaspersky 2011). Figure 1.2 is a summary of the days taken by different worms to be created, based on the day on which the vulnerabilities were released.

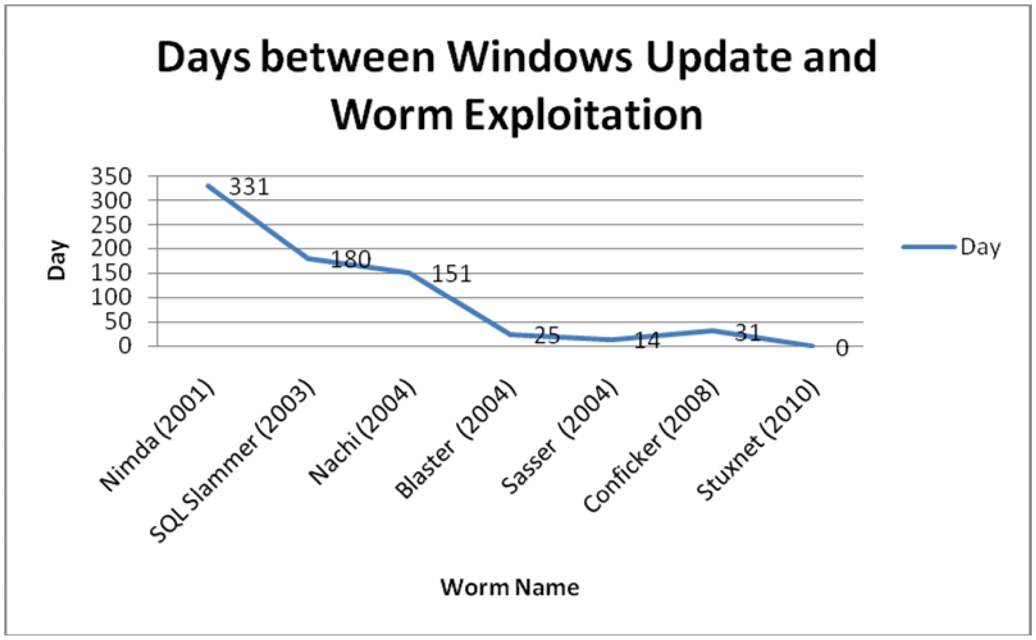


Figure 1.2. Timeline for Worms' Exploitation.

It can clearly be seen that the days needed for the worms to be created were reduced tremendously from 2001 to 2004. It took an average of 23 days for these worms to exploit these vulnerabilities during the period 2004 to 2008. The Conficker worm is one good example of a worm that exploits Microsoft's vulnerability in the victim's computer. Most worrying is the Stuxnet worm since this worm exploits the vulnerability in Windows before the software developers were aware of it and this kind of exploit is known as a 'zero-day' vulnerability (Kaspersky 2011, Bradley 2010). Apart from this, social network sites such as Facebook, MySpace, Twitter and Buzz have also been misused, and have been targeted by worms such as Myspace XSS worm in 2005 (Laborger 2005), followed by the Koobface worm in 2008 (Vamosi 2008) and the XSS exploits in 2009 (Robert 2009). These worms have been successful due to the vulnerability of these sites. Shepherd (2003) explains that this vulnerability as a form of a flaw which has later been exploited to allow unauthorized access, elevation of privileges and denial of service (DoS). To a certain extent, even though the vulnerability has been patched, a worm can still infect a victim's computer via USB and shared folders with weak passwords. Therefore, it is suggested that the incident response is applied and integrated in analysing the way the worm infects and propagates.

One of the most common mistakes made by an end user or by an organisation when dealing with a worm infection is not following the right procedure or steps for eradication (White and Granado 2009). Following the standard operating procedures is the heart of an incident response. Thus it not only helps to reduce response time, but also reduces the financial and productivity loss due to a worm incident. In an incident response, the main aim is to recover quickly and effectively from a security incident, to respond systematically by following standard procedures, and to minimize the impact caused by disruption in critical computing services (Schwetzer 2003). It has been estimated that the average total cost for large organisations for computer security incidents is between £280,000 and £690,000 (Pricewaterhouse Coopers 2010). Moreover, according to this report, there were a few changes in the security landscape during 2010. This is shown diagrammatically in Figure 1.3.

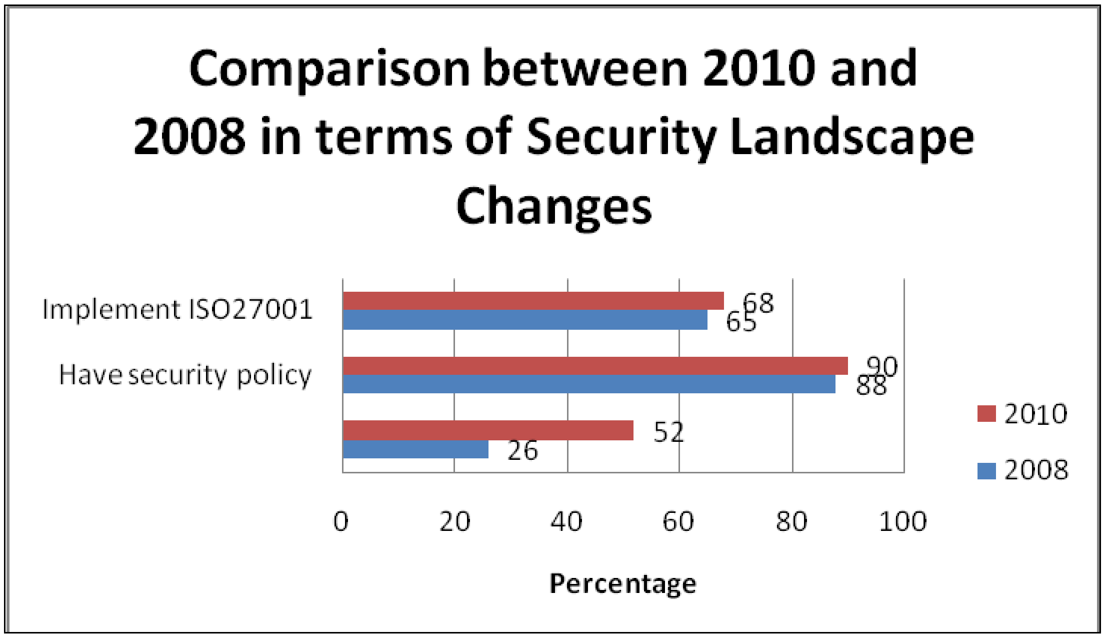


Figure 1.3. Comparison between 2008 and 2010 for Security Landscape Changes. Adapted from Pricewaterhouse Coopers 2010 Information Security Breaches Survey (Pricewaterhouse Coopers 2010).

Based on Figure 1.3, there was an increment of 26% in the year 2010 compared to year 2008 for ongoing security training provided by organisations for their staff. Apart from that, the implementation of security policies and ISO 27001 indicates the growth of security awareness among users.

For worm detection, it is noticed that most of the work carried out such as that by Schultz and Shumway (2001) and Henchiri and Japkowicz (2006) particularly focused on the features of worm detection. In contrast to this thesis, this thesis explored in more depth the worm threats and architecture which is later used as the input to develop a better method and model for worm detection and response. While in terms of the incident response perspective, there are a few studies related to incident response in general (Mitropoulous *et al.* 2006;

Goel and Gangolly 2007) none explain in detail the incident response that is associated with worm detection and response.

Also in 2011, Bejtlich *et al.* (2011) raised an issue related to the incident detection and response team about using the traditional incident detection response model. Bejtlich *et al.* (2011) claimed that this traditional model should be improved since it failed to identify application-level compromises. Moreover, Bejlicth *et al.* (2011) said that nowadays intruders are more concerned with accessing data rather than owning the computer. They also list four main problems with regard to the incident response team. These are: firstly, a lack of understanding of the different applications; secondly a lack of understanding of the subtle activities undertaken by the intruder with regard to the applications; thirdly a lack of understanding of different instrumentation used to interpret the logged data; and lastly a lack of knowledge about how to interpret the logged data. One of the biggest mistakes made by most security analysts was to assume that all intrusion issues can easily be solved by using network appliances to sniff the network for any abnormal activities. To a certain extent, these devices cannot detect the abuse and misuse at the application level, where clear information about user actions and data access is really hard to monitor (Garfinkel and Rosenblum 2003).

In conjunction with the challenges raised by Bejlicth *et al.* (2011), a solution is needed to help users to confront the worm infection while waiting for the incident response team to offer its analysis and help. The solution or model developed must evolve with time, and must be capable of isolating a worm infection from propagating further, so later any new method integrated by the worm especially the polymorphism and stealth worm can be detected and isolated easily. At the same time, it must also help the user to detect and respond to the worm incident. Based on this thesis studies and observations, one of the most promising approaches to responding and isolating a worm infection is by using the apoptosis approach, which has been tested and showed an encouraging result (refer Chapter 5,section 5.4.2). Apoptosis is one of the specialisms found in human immunology and is known as cell-programmed death. It has the ability to kill itself once it has killed the intruder. Apoptosis is explored and integrated into this thesis. From a worm response perspective, the apoptosis disconnects the infected computer from the Internet or network once it has identified the victim's computer is in a dangerous position. Here the challenge is how to decide when the apoptosis should be triggered. This thesis claims to have successfully solved this challenge and arrived at a solution whereby weight and severity are the most important factors to trigger the apoptosis condition. Indeed, the STAKCERT worm apoptosis algorithm has been developed to solve this challenge.

Based on all the motivation issues discussed here, this thesis intention is to make significant and new contributions to the security field, especially in the worm detection and response field. This particular area of research will therefore be focused on.

1.3 Research Aims and Objectives

The main aim of this research is to come up with a new model for worm detection and response by integrating the best techniques from incident response, knowledge discovery, security metrics, apoptosis and data mining, with the goal of creating a new model which is more effective than the existing ones. The scope of the research is on a Windows platform only. Prior formation of the new model, a thorough and in-depth study on the worm architecture, worm classification, worm analysis, worm detection techniques and worm response techniques were carried out to obtain a better understanding of the underlying methods and techniques in the existing works, so that any related improvements can be done along the way.

The objectives of this research are:

- To conduct an in-depth study of worm architecture and worm classification and to introduce a new worm classification and relational model.
- To conduct an in-depth study of the existing methods of worm analysis and to introduce a new technique to improve worm analysis techniques.
- To improve the existing worm detection techniques to give a better accuracy.
- To introduce a new technique of applying apoptosis to worm response to avoid the worm from further propagating.

- To conduct an in-depth study of the requirements needed to trigger the apoptosis and to formulate new algorithm with regard to how to trigger the apoptosis.
- To provide an accuracy rate for the STAKCERT model for worm response which can be used as a comparison by other researchers in future.

1.4 Research Contributions

The contributions for this research are:

- A STAKCERT model for worm detection which consists of:
    - STAKCERT worm classification.
    - STAKCERT worm relational model.
    - Enhanced STAKCERT KDD processes for worm analysis which is the integration of static analysis, dynamic analysis, statistical analysis and standard operating procedures with regard to incident response.
- An improved overall accuracy for worm detection rate as a result of using the STAKCERT model compared to existing works.
- A STAKCERT model for worm response which consists of:
    - A new technique to respond to worm infection by applying apoptosis.

    - A STAKCERT worm apoptosis algorithm on how weight and severity rank and value are assigned to trigger the apoptosis based on security metrics.
- An accuracy rate for worm response as a result of using the STAKCERT model which can be used as a comparison by other researchers in future.

To support the contributions listed above more, reviewers' comments from an internationally recognized conference, publisher and local seminars have been taken into consideration. Improvements have been made to ensure the quality of the research and to ensure that this research has made a significant contribution. A list of the publications related to this research can be seen within the research publications section.

1.5 Thesis Organisation

The rest of the thesis is structured as follows:

Chapter 2 contains a literature review where related studies and the fundamental knowledge of the subject matter are discussed. This includes the worm study, which consists of the definition, comparison with viruses and Trojan horses, worm classification, incident response and apoptosis. Existing works related to this research are also explained here.

Chapter 3 discusses in detail the research methods and the performance criteria used for this research. The STAKCERT KDD processes for worm analysis and response which is the integration of static analysis, dynamic analysis, statistical analysis, security metrics, data mining and standard operating procedures in terms of incident response is also presented in this chapter.

Chapter 4 explains of the STAKCERT model for worm detection in detail. It consists of the experimental results, the STAKCERT worm classification and the STAKCERT worm relational model. Different testing techniques and a comparison with existing work was conducted to prove the effectiveness of the STAKCERT model for worm detection. Here the limitations of this thesis are included.

Chapter 5 discusses the STAKCERT model for worm response in detail. It consists of the experimental results and the STAKCERT worm apoptosis algorithm which apply weight and severity. Different testing techniques and a comparison with existing works were conducted to prove the effectiveness of the STAKCERT model for worm response. Here the limitations of this thesis are included.

Chapter 6 concludes the research by summarizing and discussing the contributions made and future work.

1.6 Summary

There is an urgent need to produce more research about worms. Worms are always seen as one of the main threats in the cyber world and it is a topic which has been discussed ever since the first worm was invented. The motivation to pursue research in this area is to solve the challenges to detect and respond effectively to the threat of worms. Based on these thesis objectives and the contributions made, this thesis can be used as a guide and basis for further exploration for other researchers all over the world who have the same interests.

CHAPTER 2

LITERATURE REVIEW

Chapter 2 contains a review of the literature where related studies and the fundamental and core knowledge of worms and apoptosis are discussed. This includes worm studies which consist of definitions, comparison with other malicious code categorizations, worm classifications and worm detection and response techniques. Furthermore, apoptosis is discussed because of its key role in terms of worm response in this research. Nevertheless, previous works that are related to this research are presented and studied in order to deal with the gaps found in these previous pieces of research.

2.1 Worms

In this section, apart from the definition and comparison of worms, previous work that is related to this thesis is presented, in terms of worm classification, worm detection and response techniques.

2.1.1 Definition

A clear definition of a worm is a must, as is an understanding of the worm's architecture. According to Nazario *et al.* (2001), a worm is defined as an independent replicate, and autonomous agent, that is capable of searching through the network for a new host system which it may then infect. The structure of a worm is divided into six main components. These are reconnaissance capabilities, special attack capabilities, command interface, communication capabilities, intelligence capabilities and unused attack capabilities. Nazario *et al.* (2001) claimed that a divided worm structure eases the process of worm detection and prevention. Helenius (2002) defined a computer worm as an independent program that can replicate recursively by itself. He classified malicious code in accordance with their characteristics and the infected objects. Skoudis and Zelster (2004) provided a further definition of a worm where they defined it as a self-replicating piece of code that spreads through networks and does not need help from human interaction in terms of propagation.

For the purposes of this research, a worm is defined as a malicious program that can replicate itself, moving from one computer to another or can propagate via a network without human intervention or an owner's consent. A worm may be further classified into a host or a network worm.

Ellis (2003) defines a network worm as program which has the capability to execute a copy of itself in a remote computational computer and which can evolve. Meanwhile FitzGerald (2008) defines a network worm as a program which consists of several segments, with each segment running on different computers and using different networks for communication purposes. As for the host worm, it would infect a computer and remain inside that computer. It would then use the network connection to spread itself to other computers. It would kill itself after it has replicated itself to other computers (FitzGerald 2008). Table 2.1 is the comparison between a host and a network worm based on the analysis conducted in this thesis.

Table 2.1. Comparison between a Host Worm and a Network Worm.

Host Worm	**Network Worm**
1) It is self replicating and self-contained.	1) It is self replicating and self-contained.
2) It consists of one segment and uses networks to spread itself only. Can also spread via other media such as removable drives.	2) It is composed of multiple segments with different functions. It has one main segment which coordinates with other segments in the different infected computers. It uses networks for different purposes such as for communication and propagation.
3) Once it infects a computer, it runs on it and remains in the infected computer. If it propagates and infects a new computer, it replicates itself identically and terminates the original worm.	3) Once it infects a computer, it runs on it and tries to spread itself through networks as much as possible to a new computer. It will not terminate the original worm and can evolved.

There are many other worm categorizations that can be further classified under host and network worms. Cohen (1992) referred to Internet worm attacks as being totally dependent on specified bugs or vulnerabilities in a victim's computer, whereas it does not evolve and replicate itself identically. Examples of Internet worms are the Nachi Worm, SQL Slammer, Nachi and Conficker worms. These worms exploit vulnerabilities found in a victim's computer.

On 10th July 2009, distributed denial of service (DDoS) attacks took place in Korea and the USA. These caused severe damage to many organisations in these two countries. Not only was there a loss of money, due to data being corrupted, but they tarnished the reputations of many organisations, especially organisations which were involved in online banking. It caused network operation to be intermittent in certain organisations and banks in particular were heavily targeted by the attackers. In addition, in Malaysia on 13th April 2011, the Malaysiakini - one of Malaysia's top political news portal - also became the victim of the DDoS. This was believed to be linked to the hotly contested state elections on Borneo Island.

The DDoS attacks were possible due to several factors. The easiest way to conduct a DDoS attack is by exploiting vulnerabilities in the victim's computer. Vulnerabilities can be exploited using framing code or by overwriting the normal computer program. The DDoS attacks use certain ports to launch the attack. The attack launched can be categorized as an internet worm as it exploits the vulnerabilities of applications. The Kaspersky website (Kaspersky 2008) defines internet worm as a program which distributes itself in many ways, and one of these ways is by exploiting operating system vulnerabilities. It is still considered to be a host worm as it remains inside the computer and uses internet connections to spread itself to other computers.

2.1.2 Comparison With Other Malicious Code

The Internet is constantly being flooded with information about computer virus, worm, trojan horse, adware and spyware. These terms have been used interchangeably, but most of the time the public do not know that they have different meanings and functions. Thus, it is critical to understand this malicious code or what is called as computer virus, worm, trojan horse, adware and spyware, to ensure the detection and response techniques that will be applied are suitable based on its characteristic. Malicious code can be referred to as any program that moves from one computer to another or from network to network, and can modify a computer system without the consent of the owner or the operator (Kienzle and Elder 2003). There are many ways in which malicious code spreads. The common media are through email attachments, scripts in web pages and network and file sharing. In this research, this thesis specifically focus on worm.

Cohen (1985) first introduced the term of computer virus in 1983 and formally defines the computer virus definition in 1987 (Cohen 1987). Furthermore in 1989, Adleman (1989) proposed another computer virus definition which was based on set theory. Basically these previous works, define a computer virus related with a broad range of replicating programs. Therefore, based on the experiment and analysis conducted, a computer virus is defined as a program which when executed can add itself to other programs, without permission or right. This is done in such a way that the infected program, when executed, can add itself to other programs as well. The computer virus inserts itself into the chain of command and executes a legitimate program that results in the execution of the computer virus together with the program. If relation is made to human daily lives, computer virus programming logic mimics its human virus biological counterparts. First, it invades the victim’s host by changing the underlying structure. Once infected, host files become viruses themselves and begin to infect other files. Later, computer viruses mutate and evolve to fight anti-virus programs, and this massive infection results in the larger systems malfunctioning.

On the other hand, based on the analysis carried out, a worm is defined as a program that replicates itself from one computer to another and does not need a host file to spread itself (Weaver *et al.* 2003). This is in contrast to a computer virus which requires a host file to spread itself. As for a trojan horse, this is a malicious program which tricks users into believing that it is a genuine file. It must be executed in the victim’s computer and once this has been done, it can control the victim’s computer remotely and steal any confidential information from it. Apart from that, the trojan horse does not replicate itself. As for adware and spyware, they can easily be installed on a user’s computer when the user downloads free software or browses the Internet. Adware usually comes together with free software or with a demo version of software. Generally, all settings in free software are enabled by default. Therefore, the user must be aware of the end user license agreement (EULA) which is provided. Sometimes, this software comes together with advertisements or add-on tools and is installed automatically together with the free software, without the user’s knowledge. It is highly advisable only to install software from a trusted source, since there is the possibility that adware will be installed together with the downloaded software. The adware tracks the user’s surfing habits so that later it can serve related advertisements to the user. To a certain extent, it might try to steal the user’s username or password, monitor user activity on the Internet, gather information about e-mail addresses and credit card numbers and transmit them to someone else. Once it becomes intrusive, it is then categorised as spyware, and it should be avoided for privacy and security reasons. Spyware is considered as a malicious program and is similar to a trojan horse when the user unintentionally installs it together with the genuine program.

The differences between computer virus, worm, trojan horse, adware and spyware is summarised in Table 2.2.

Table 2.2. Comparison between Worm and Other Malicious Code.

Computer Virus	**Worm**	**Trojan Horse**	**Adware**	**Spyware**
1. Non self-replicating.	1. Self-replicating.	1. Non self-replicating.	1. Non self-replicating.	1. Non self-replicating.
2. Produces copies of itself using host file as carrier.	2. Does not produce copies of itself using host file as carrier (independent program).	2. Does not produce copies of itself using host file as carrier (independent program).	2. Produces copies of itself using host file as carrier.	2. Does not produce copies of itself using host file as carrier (independent program).
3. Cannot control PC remotely.	3. Cannot control PC remotely.	3. Can control PC remotely.	3. Cannot control PC remotely.	3. Can control PC remotely.
4. Can be detected and deleted using anti-virus software.	4. Can be detected and deleted using anti-virus software.	4. Can be detected and deleted using anti-virus and anti-rootkit software.	4. Can be detected and deleted using anti-virus and anti-adware software.	4. Can be detected and deleted using anti-virus and anti-spyware software.

2.1.3 Worm Classification

Based on the experimentation and analysis conducted, it is suggested that a comprehensive structure of a worm classification which considering characteristics in worm, can be used as the basis for a worm detection and response technique. Therefore, a STAKCERT worm classification is produced based on the testing and comparison associated with research by Dabirsiaghi (2008), Nazario *et al.* (2001), Helenius (2002), Skoudis and Zelster (2004) and Saudi *et al.* (2008a). The STAKCERT worm classification is based on and was formed in accordance with five main attributes. These are infection, activation, payload, operating algorithm and propagation. This is explained in detail in Chapter 4, section 4.3.1. Based on the experimental results for worm detection and response which were explained in detail in Chapters 4 (section 4.4) and 5 (section 5.4), the STAKCERT worm classification used in this thesis helps to increase the accuracy rate compared with existing methods.

2.1.4 Worm Detection and Response Techniques

Once an understanding of the worm architecture is gained, a further issue is considering different perspectives and analyzing the gaps, drawbacks and challenges that should be taken into consideration in producing effective worm detection and response techniques. White (1998) discussed problems within the research field of computer worms, such as heuristic techniques, the epidemiology of worms, the digital immune system, technology in dealing with worms and proactive approaches to controlling them. Much more research has been carried out since then, in order to address the problems and challenges raised. Filiol *et al.* (2006) discussed open problems within computer virology, claiming that only a limited number of studies had addressed computer virology.

Much research has been conducted in the past few years related to worm detection such as that by Henchiri and Japkowicz (2006) who were able to increase the accuracy of a virus classifier using the N-gram method compared to the approach used by Schultz *et al.* (2001) where they used the same dataset, Tseng and Lin (2009) who used the ‘variant objects discovering acquisition’ (VODKA) method as a basis for detecting worms and Agosta *et al.* (2007) who used an adaptive end-host anomaly detector to detect worms. Meanwhile, Moskovitch *et al.* (2008a) conducted experiments using different techniques of machine learning to detect worms based on computer behaviour, and identified Bayesian Networks as the best algorithm. In addition, Siddiqui *et al.* (2009) used the static features of a worm program to detect worms. Dai *et al.* (2009) incorporated dynamic instruction sequence mining techniques involving the runtime features of a worm program to detect worms and Stopel *et al.* (2009) used artificial neural networks (ANN) to detect worms. Each of these works has it owns strengths and gaps that can be further improved. Therefore, based on this thesis analysis and review of the existing works, it is suggested that one field that lacks research and thus needs to be explored in more depth is incident response.

Incident response is defined as the process that aims to minimise the damage caused by security incidents and malfunctions. It also monitors and learns from such incidents (BSI 1999). The lack of standard operating procedures, in terms of analysing and responding to a worm infection, may lead to disaster for both IT personnel and the end user. It is very hard to separate incident response from the worm detection and response field, as it plays a very important role within such a field. Improvements and novel standard operating procedures, particularly within the detection, analysis and disinfection phases, are seen as areas for potential research and exploration (Werlinger *et al.* 2010).

Examples of work related to incident response are those by Mitropoulos *et al.* (2006), Goel and Gangolly (2007), Vasudevan (2008), Kim *et al.* (2010) and Liu *et al.* (2010). An example of research that proposed a generic incident response process within a corporate environment is that undertaken by Mitropoulos *et al.* (2006) in 2006. A year later, Goel and Gangolly (2007) proposed a two tier model for handling malicious code. This work integrated an immunology and an epidemiology approach in conjunction with a distributed database. At the same time, work by Vasudevan (2008) claimed that the system known as MaiTrak was capable of tracking malware without using any signature base, eliminating the malware and returning it to its prior clean state. In the case of Kim *et al.* (2010), they proposed an incident response system based on DSS framework which applied Recency, Frequency and Monetary (RFM) analysis methodology and case-based reasoning (CBR), whilst Liu *et al.* (2010) designed a system using an ontological approach and CBR. Both of these works have their own approaches to detect and response to the incident. However, based on these previous works, research alluding to a combination of worm handling procedures following incidence response has, so far, been scarce. It is suggested here that such research could greatly improve matters by detailing the required procedures for handling a worm incident. This is one of the precepts of the formation of the STAKCERT model for worm detection, of which incident response is a part.

2.2 Apoptosis

In this section, apoptosis is defined and a comparison between apoptosis and worm problems is conducted. Apart from this, this section presents previous work that is related to this thesis.

2.2.1 Definition

The human body is divided into many compartments which provide robust security against intruders. These small independent compartments are called cells, and they are the basic building blocks of the human system. Each cell controls what may enter and exit its membranes, keeping the internal organelles protected (Purchon 2000; Sullivan 1994). Individual cells are disposable, so the death of one cell does not affect the entire person. Humans live in an environment where humans are constantly being attacked by intruders such as viruses, bacteria and other organisms, yet the majority of humans survive these attacks for many decades. It is not necessary to download any security patches since human bodies have adapted to living in such a harsh environment. To improve the computer systems survivability, the biology field offers a great deal of opportunity for further exploration and integration within the computing area (Somayaji *et al.* 1997). Based on the analysis and experimentation conducted in this thesis, apoptosis is seen as one of the specialisms in human immunology that can be further explored and integrated into this thesis, particularly in response to worm infection.

Apoptosis or cell-programmed death is a highly regulated process that allows a cell to self-degrade in order to eliminate unwanted or dysfunctional cells from the body. According to Ishizaki and colleagues (2005), apoptosis is the process by which cells die as a natural course of events. It is also means ‘drop out’, and was used by the Greeks to refer to the shedding of leaves by trees in the autumn; i.e. the loss of cells that ought to die in the midst of the living structure. The process has also known as ‘death by default’, where cells are prevented from putting an end to themselves due to the constant receipt of biochemical ‘stay alive’ signals. Furthermore, Martin (1998) stated that during apoptosis, the genome of the cell fractures, the cell shrinks and part of the cell disintegrates into smaller apoptosis bodies. It is a controlled process whereby the content of the cell is kept strictly within the cell membrane as it is degraded. According to Martin, the cell is phagocytosed by macrophages before its contents have a chance to leak into the neighbourhood. At this point, the apoptosis helps to prevent an unwanted inflammatory response. Besides, apoptosis is essential to embryonic development and to the maintenance of homeostasis in multicellular organisms. In the immune system, cell death eliminates B cells and T cells that elicit autoimmune responses, and selects the most efficient lymphocytes to encounter antigen in the process of affinity maturation. Cell death can occur in two ways; necrosis and apoptosis, and although both terminate in cell death, the intracellular pathways of each process are very distinct. Necrosis involves the unregulated death of a cell following cell stress, and results in total cell lysis and subsequent inflammation due to the existence of the cell debris. On the other hand, apoptosis is a regulated form of cell death with defined intracellular pathways and regulators (Kerr *et al.* 1994). Furthermore, apoptosis is used to destroy cells that may be a threat to the organism such as cells infected with a virus, cells with DNA damage and cancerous cells. These cells can be disposed of without causing harm or stress to other cells. This underlying apoptosis concept is the one that is integrated into this thesis.

2.2.2 Applying Apoptosis in Worm Response

From a worm response perspective, the apoptosis concept is mapped into the computing environment by disconnecting the severely infected computer from any network to avoid the worm in the infected computer from further propagating to other computers in the same network. If in the human immune system, intracellular pathways and regulators need to be defined, this is also applied in the computing area where rules and procedures were defined to trigger the apoptosis. So later when the apoptosis is being mapped in the worm response perspective, a clear understanding on what are the factors that should be taken into consideration to trigger apoptosis can be easily identified. In responding to worm infection and deciding on the apoptosis condition, a

STAKCERT worm apoptosis algorithm, which is explained in detail in Chapter 5 (section 5.3.1) is developed. Table 2.3 shows the comparison between apoptosis and worm problems. The apoptosis is mapped into the STAKCERT model for a better understanding of the apoptosis concept.

Table 2.3. Comparison between Apoptosis and Worm Problems.

Apoptosis	**Worm Problems**
Once a cell is infected by an intruder, the cell tries to recover from the intruder infection. If the cell cannot recover from the infection, instead of spreading the infection to other cells, it kills itself. This process is known as apoptosis. Indeed, if the formation of the cell is abnormal, the apoptosis is triggered as well.	Worm infects a computer in many ways such as through email and USB. The apoptosis concept in computing helps to prevent the worm from further propagating. This is done by recognizing a computer which is severely infected by a worm and distinguishing this from one that is not severely infected by a worm. In the STAKCERT model, a computer that is severely infected by a worm is identified, based on five main worm attributes. These are payload, propagation, activation, infection and the operating algorithm. The STAKCERT worm apoptosis algorithm is developed to show how the weight and severity value are assigned for the five main worm attributes.
In biological systems, all cells share the same apoptosis mechanism, hence cell suicide is essentially the same process no matter what kind of cell it is.	Computers, however, play very different roles in the IT structure. This dictates different ways of dealing with the need to sacrifice a particular computer for the good of the system as a whole. Dealing with an infection in a key database server is a far more delicate operation than dealing with an infected perimeter computer, especially a PC, PDA or handphone. In the STAKCERT model, a severely infected computer is disconnected from the network.

Apoptosis provides lots of opportunity for exploration to be implemented or integrated in the computer security field. It started with research by Tschudin (1999) where he discussed the opportunity for integrating apoptosis into distributed mobile services. He also discussed security issues such as how to secure apoptosis. This paper can be used as the basis for forming other security tools. The concept of apoptosis was implemented in different applications with different goals. Examples of such works are those of Riordan and Alessandri (2000) who used apoptosis to shut down certain services in Windows within a computer, once the computer has been identified as vulnerable and had the potential to be exploited. Meanwhile Olsen *et al.* (2008) built the HADES system, which included the programmed death concept as one of its methodologies. In this system, agents were primarily used for communication, and had the authority to do repairs and undertake regeneration, movement and death (programmed death). This system relied totally on the existing agents and flaws might have arisen due to irregular agent mutation. In 2008, Saudi *et al.* (2008a, 2008b) explained how apoptosis could be integrated into computer security and integrated apoptosis in an intrusion detection system (IDS). However, this paper focused only on IDS. The challenge would be to implement the idea in other security tools. Furthermore, Mulholland *et al.* (2008), introduced a roadmap for the design of a tagging and tracking system for data security used in prisons and correctional facilities, whilst Tarakanov (2008), introduced an intelligent intrusion detection system using apoptosis as part of the system. In addition, Ben Othmane and Lilien (2009) introduced 'Active

Bundles' to protect data security. Moreover Hively *et al.* (2010) wrote a paper that explained briefly that apoptosis could be mapped easily in a cyber security analogy by terminating access to the network when there was any sign of unauthorized activity or a violation of security policy. Finally, Sterritt (2011) has published a few works related to autonomic computing since 2004, and since then has integrated apoptosis in autonomic agent and swarm space exploration systems. He also wrote about the potential exploration in future regarding autonomic computing and apoptosis.

Based on all the previous works discussed above, the main challenges which should be considered thoroughly are the method of assigning apoptosis and the scope for its implementation, where there is still a lack in terms of responding to a worm incident. Therefore, based on the experiments and analysis conducted in this thesis, weight and severity are identified as two important factors which trigger apoptosis. The STAKCERT model proposed not only focuses on worm response, but it also focuses on worm detection, which has also been integrated and considered in this thesis. Further details of this STAKCERT model can be referred in Chapter 4 and Chapter 5.

There are a few studies which have considered weight as part of their work. Examples are those of Su (2011), who built a real time anomaly detection system for denial of service (DoS) attacks using weighted k-nearest neighbour classifiers, Siddique and Maqbol (2011), who used weighting in software clustering, Kim *et al.* (2010), who used weight as part of the log analysis of incident response in a DSS system, Fisch *et al.* (2010), who used weight to optimise radial basis function neural networks for an intrusion detection system and Middlemiss and Dick (2003), who used weighted feature extraction using a genetic algorithm for an intrusion detection system. Based on these works, it can be concluded that there is no standard way of assigning weight, which has been seen as an important feature in increasing the accuracy, or optimising the performance, of different works in different fields. Therefore, weight is integrated within the STAKCERT model and used security metrics and frequency analysis to retrieve the rank and the value of the weights. Later, the weights are used for assigning the level of severity which triggers apoptosis.

In a study conducted by Miles (2001), he assigned a severity incident into three categories which are high, medium and low. The high severity involves incidents with long term effects to the business or critical system i.e root access, denial of service (DoS) and it also involves with unauthorized privilege (root), limited access (user), unsuccessful attempt, utilisation of services and probe, poor security practices, malicious logic, hardware, software or infrastructure failure and espionage. Medium severity involves non-critical system and detection of initial attack and low severity involves detection on reconnaissance, threats of future attacks and rumours of security incidents.

In an open source tool, for example Nessus, the severity is assigned in six levels which are none, low, medium, high, serious and critical. ‘None’ represents no risk, ‘low’ as useful information to an intruder i.e software versions and ‘medium’ stands for the existence of a security hole that can lead to privilege escalation. As for ‘high’, it enables the attacker to gain administrator privilege, ‘serious’ means the attacker can gain profit from the confidentiality information retrieved using the administrator or user privilege and ‘critical’ means the victim’s host already belongs to the attacker.

On the other hand, Reese (2003) defined high severity as posing a threat to an entire autonomous system, such as a university network; that is a threat to the operation of critical network systems that threatens one or more applications that are integral to daily university functions. Medium severity involves a risk to isolated and non-production university systems and low severity involves minimal exposure of threats. Indeed, the University of Florida (2010) has taken the same initiative by dividing the incident severity to high, medium and low. High severity involves data security on the critical data i.e bank account, intellectual property, legal issues where it might cause loss of money more than USD10,000, child porno, copyright violations, magnitude critical service disruption, more than 10% of network asset infected, attacking other computers and public interest. Whilst for medium severity involves data security on sensitive data i.e non-personal data, legal issue with money loss less than USD10,000, harassment, 3% to 10% of network assets infected, active attacking from inside a computer and public interest. Low severity involves other than high and medium severity contents.

By referring to the previous studies conducted in assigning severity, this thesis came out with a conclusion that severity must consider the data criticality, infrastructure availability and loss of productivity where these have been integrated as part of the security metrics. These three factors are mapped in security metrics, which can be seen in Chapter 3 (section 3.2.4.3).

The STAKCERT model attempts to fill in all the gaps and challenges from the previous research to detect and respond to a worm. This thesis aims that in the future, this model will be implemented as worm detection and response software.

2.3 Summary

In this chapter, literature reviews were presented on the underlying fundamental and core knowledge required to undertake this research involving worm and apoptosis studies. The literature review began with a discussion on worm studies which consisted of definitions, classifications and a consideration of worm detection and response techniques. Subsequently this was followed by a consideration of apoptosis studies which consisted of definitions and how apoptosis is applied in different fields. Apart from that, all works related to these two core areas of knowledge were also presented in this chapter. The results in terms of the improvements made are based on the gaps identified in the related works and are presented in Chapters 4 and 5.

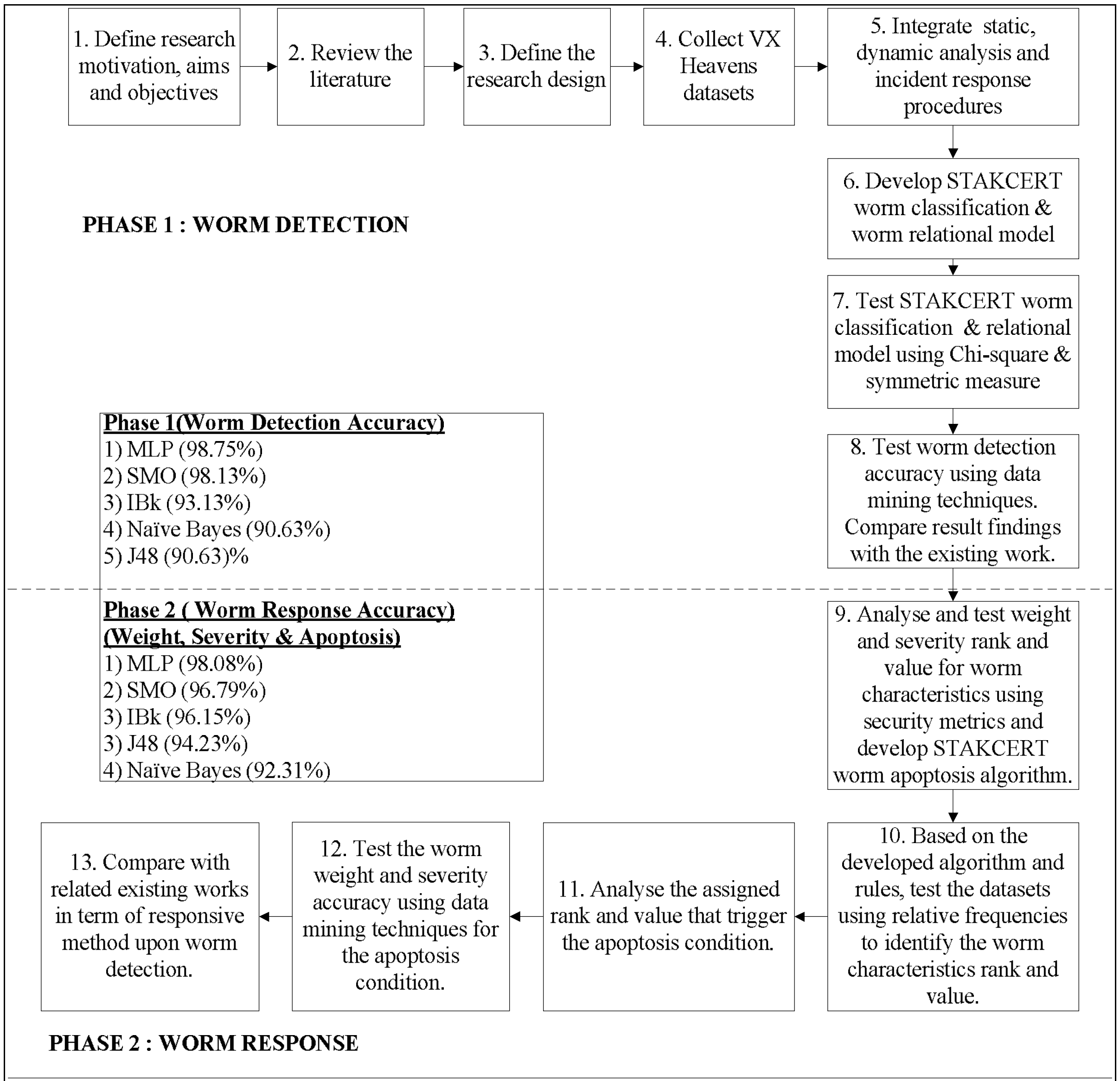
CHAPTER 3

STAKCERT RESEARCH METHODOLOGY

Chapter 3 explains the STAKCERT research methodology including detailed explanations of what methods have been used to collect and analyse the data, how the research has been conducted, why these methods were chosen and how the findings from this research have been tested and verified. A good quality research finding comes from a well structured and systematic research methodology, where there are always answers to any questions regarding the research and it is replicable by other researchers. The results from a thorough methodology will be rigorous.

3. 1 Overall STAKCERT Research Processes

Towards this end, this research proposes a model known as the STAKCERT model for worm detection and response. All the processes involved in forming the STAKCERT model are simplified in Figure 3.1. There are two phases involved in this research, which are the worm detection (Phase 1) and worm response (Phase 2).


1. Define research motivation, aims and objectives
2. Review the literature
3. Define the research design
4. Collect VX Heavens datasets
5. Integrate static, dynamic analysis and incident response procedures
6. Develop STAKCERT worm classification & worm relational model
PHASE 1 : WORM DETECTION
7. Test STAKCERT worm classification & relational model using Chi-square & symmetric measure
Phase 1(Worm Detection Accuracy)
1) MLP (98.75%)
2) SMO (98.13%)
3) IBk (93.13%)
4) Naïve Bayes (90.63%)
5) J48 (90.63)%
8. Test worm detection accuracy using data mining techniques. Compare result findings with the existing work.
Phase 2 (Worm Response Accuracy) (Weight, Severity & Apoptosis)
1) MLP (98.08%)
2) SMO (96.79%)
3) IBk (96.15%)
3) J48 (94.23%)
4) Naïve Bayes (92.31%)
9. Analyse and test weight and severity rank and value for worm characteristics using security metrics and develop STAKCERT worm apoptosis algorithm.
10. Based on the developed algorithm and rules, test the datasets using relative frequencies to identify the worm characteristics rank and value.
11. Analyse the assigned rank and value that trigger the apoptosis condition.
12. Test the worm weight and severity accuracy using data mining techniques for the apoptosis condition.
13. Compare with related existing works in term of responsive method upon worm detection.
PHASE 2 : WORM RESPONSE

Contribution (phase 1 : Worm detection):

1) To develop a new STAKCERT model for worm detection:

A) Consists of STAKCERT worm classification.

B) Consists of STAKCERT worm relational model.

C) Consists of an enhanced STAKCERT KDD processes for worm analysis algorithm which is the integration of static analysis, dynamic analysis, statistical analysis and standard operating procedures in incident response.

2) To compare with the existing worm detection techniques and improved overall accuracy detection rate using STAKCERT model for worm detection.

Contribution (phase 2 : worm response)

3) To develop a new STAKCERT model for worm response:

A) Consists of STAKCERT worm apoptosis algorithm on how weight and severity rank and value assigned to trigger the apoptosis.

4)To compare with the existing work on the techniques applied for worm response and introduced new technique for worm response by integrating the security metrics and apoptosis.

5) To provide an accuracy rate for the STAKCERT model for worm response .

Figure 3.1. An Overview of the STAKCERT Research Processes.

Thirteen processes make up these phases, which start by outlining the research background. The prior formation of the STAKCERT model and the motivation, aims and objectives are well defined and focused to ensure the contribution produced at the end of this research has a significant value. Details of these can be found in Chapter 1.

Once the first process is complete, it is followed by a review of the existing literature. The STAKCERT model introduced in this research covers most of the gaps identified in an earlier study conducted by previous researchers. A depth analysis and a comparison of the previous related works, analysed in terms of methodology implemented, findings, strengths and weaknesses, can be found in Chapter 2.

In this Chapter, the research design is explained in detail. This includes how, why, what and which data are collected and explored and the techniques applied to fulfil all the objectives for this research. All the processes from number 5 to 13 show structured and systematic processes that have been conducted to help this research to succeed. Details of these procedures implementation can be read in Chapter 4 and Chapter 5.

3.2 Research Design

In this section, all the techniques and procedures applied for analysis and testing and the datasets source involved, are clearly explained.

3.2.1 Datasets

The dataset in this research consists of different types of worms and benign executables sourced from VX Heavens (2009). From 66,711 samples downloaded from VX Heavens, 5,614 were identified as worms and 331 were identified as benign executables. Details of the worm categorisation are displayed in Figure 3.2. From Figure 3.2, it can be seen that 3.97% represent the email worm, followed by 1.36% for P2P worm, 0.96% represent the IRC worm, 0.81% for the internet worm, 0.42% for the instant messaging worm and 0.86% for other worm. While the benign executables consist of Windows system executables, commercial executables and open source executables. The datasets were chosen randomly from these worm categories and benign executables and STAKCERT KDD Processes were applied to these datasets. As a result, 160 datasets which consist of variants of the Windows worm and benign executables have been used for this research.

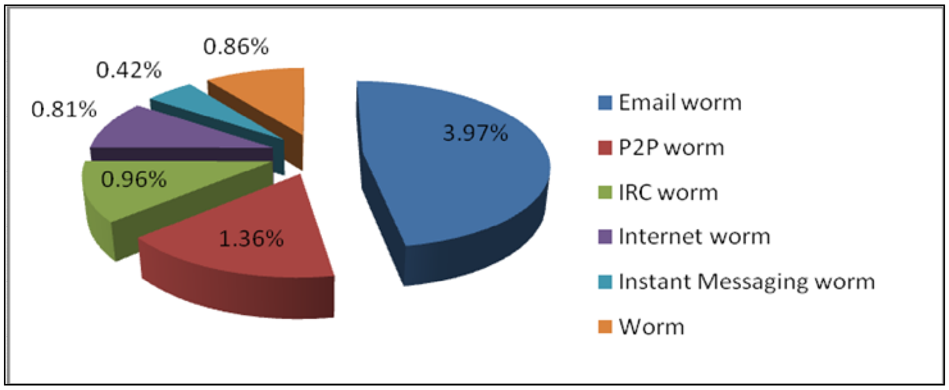


Figure 3.2. Worm Datasets.

There are several reasons why this thesis chose to gather data from the VX Heavens source; firstly, many studies have used this data for their testing, for examples, those conducted by Schultz *et al.* (2001), Henchiri, and Japkowicz (2006), Moskovitch *et al.* (2008b), Dai *et al.* (2009) and Khan *et al.* (2010). Indeed, one of the works stated above is used as a comparison with the STAKCERT research findings. The second reason is because the variants are more important than the quantity of the datasets, since these already represent different types of worm in VX Heavens and the third is due to the scope of this research, which only focuses on Windows worms. Lastly, it is one of largest worm databases freely available from the Internet.

The datasets for this research were transformed into nominal data after the static and dynamic analyses were completed, which is part of the knowledge discovery database (KDD) processes. Details of the data transformation are explained under section 3.2.3: knowledge discovery techniques.

3.2.2 Lab architecture

The lab used for this testing is illustrated in Figure 3.3. It is a controlled lab environment and almost 80% of the software used in this testing is open source or available on a free basis. No outgoing network connection is allowed for this architecture. In this lab, the data described above were tested. From these tests, the results can easily be analysed and any flaws found can be fixed immediately. For testing purposes, a checklist which consists of all the software was produced to ensure all the software was installed and working in these test lab computers. A list of the software installed in the test lab computers is displayed in Table 3.1.

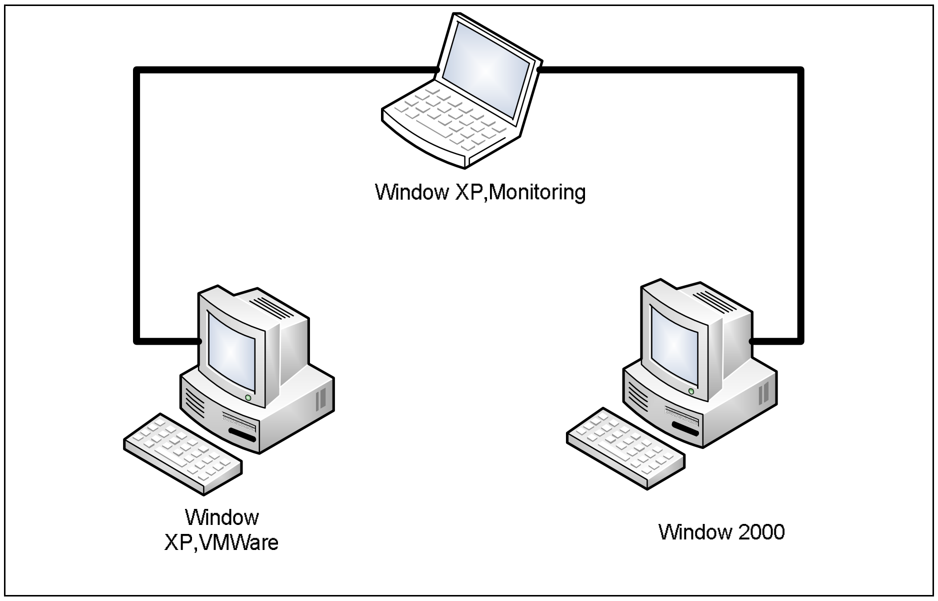


Figure 3.3. Lab Architecture.

Table 3.1. Software Installed in Testing Lab Computers.

Function	**Tools**	**Purpose of Action**
Scan tool	• TDS-3 • AVG antivirus • Ad aware	To prepare the scan tool to detect various forms of malicious code including those with newer signatures.
Strings research tool	• TDS3 • Strings.exe (from Sysinternal)	To display and extract suspicious sets of ASCII characters included in a file.
Unpack tool	• Proc dump • Unpack tool • UPX tool	To decompress and unpack the worm code.
Verification tool	• Hashtab v 2.3	To verify the CRC value of the infected file.
File Integrity Checking	• DigestIT 2004	To verify the system is in a known trusted state before the worm makes any changes.
File Monitoring	• Filemon (from Sysinternal)	To provide a dynamic update of all file system activity, indicating which processes are opening, reading and writing files.
Process Monitoring	• Prcview v 3.7.3.1 • Process Explorer(from Sysinternal)	To identify the resources used by all running processes, including DLLs and registry keys. Process explorer provides a wealth of useful information regarding how the worm is impacting upon the victim computer.
Port Monitoring	• TDIMon • PortMon • TDS-3	To see which ports are listening on the trusted system. To record all TCP and UDP activity and to see various running programs send data out through a port or receive incoming data on a port.
Network Monitoring	• NeWT • TDS-3	To look for backdoor listeners recognised by NeWT or TDS3.
Network Monitoring	• Ethereal /Winshark • Windump • Wincap	To gather all traffic going to and from the target system, using a sniffer loaded on a system other than the victim computer.
Network Monitoring	• Promiscdetect.exe	To determine if the network interface is running in promiscuous mode, gathering packets destined for all systems on the LAN.
Registry Monitoring	• Regmon (from Sysinternal)	To display real time indication of all registry activity including creating, reading and writing registry keys.
Disassembler / Debug Tool	• Ida Pro • OllyDbg	To perform detailed code analysis.
Software for data testing and simulation	• Java (WEKA) version 3.6.2 • Visual Basic 6.0 Professional	To perform data mining analysis and testing.
Virtual PC	• VMWare Work Station	To allow multiple operating systems to run on a single computer.
Database tool	• Microsoft Access 2007	To save all the databases.
Design and model system	• Rational Rose 2000 Enterprise Edition	To build a related diagram of how the model works.
Flowchart tool	• Microsoft Visio 2007	To draw a diagram.

Honeypot, Metasploit and HoneyMonkey are three examples of how worm analysis and testing can be conducted. Examples of studies using Honeypot include Levin *et al.* (2003), Dagon *et al.* (2004), Sadasivam (2005) and Spitzner (2003). The concept of the Honeypot is to allow the attacker to play around and attack the systems that consist of a few computers with different functions such as a web server and a mail server, which are purposely being left as vulnerable. However, it only allows incoming traffic to the Honeypot and disallows any outgoing traffic from the Honeypot itself. An attacker would not be able to launch any attack on other networks or systems outside the Honeypot from inside the Honeypot. A few combinations of Honeypots will form a Honeynet. The constraints of the Honeypot lie in its capability to allow incoming traffic only and in terms of portability.

Metasploit is a framework for penetration testers to discover, analyse, test and release exploits (Maynor *et al.* 2007). The only drawback that needs to be improved is the predictability of the attack, since most of the vulnerabilities in Metasploit are already well known (Jordan 2005). The other possible problem that might arise is if a new worm attacks and exploits a new vulnerability that did not yet have the payload signature in Metasploit.

As for HoneyMonkey, it detects and analyses the website that hosts the malicious code (Wang *et al.* 2005). One example of a worm that can easily infect an end user browsing a website is known as the Code Red worm. If the victim's computer is not updated with the latest Windows Update, the worm can simply launch the attack by executing itself into the victim's computer when he browses the infected website. The HoneyMonkey is definitely useful in identifying these malicious websites, but it is not suitable to be implemented in this research for this thesis, as the scope and goals are different. The same applies to the firewall and Intrusion Detection System (IDS), which are dedicatedly built to detect the worm attacks.

The main reasons why this controlled lab architecture was used are, firstly: any worm infection, propagation, operating algorithm, activation and payload can be monitored without any constraint in terms of network connectivity and secondly: in terms of the portability of the lab, where the lab can be moved with ease. Thirdly: the controlled lab environment would not cause any harm to the place where the experiment was conducted, since the lab was separated from the operational network.

3.2.3 Knowledge Discovery Techniques

The phrase KDD was first discussed in a KDD workshop in 1989 (Piatetsky-Shapiro 1991) and ever since the KDD has been successfully applied in different domains all over the world. Knowledge discovery in databases (KDD) is defined as an overall process where knowledge or patterns from data are extracted, where the patterns extracted must be valid, useful and understandable. Data mining is a specific algorithm to extract the pattern from the data, which is a part of the whole KDD process (Fayyad *et al.* 1996 and Maimon and Rokach 2010). Many studies that integrate KDD have been conducted over the past few years and current research in the year 2010 include Lavrac and Zupan (2010) in medicine, Kovalerchuk and Vityaev (2010) in financial applications, Singhal and Jajodia (2010) in intrusion detection and Thearling (2010) in customer relationship management (CRM).

For this research, KDD is used as a technique to identify the worm patterns in the datasets. All of the KDD processes are summarised in Figure 3.4.

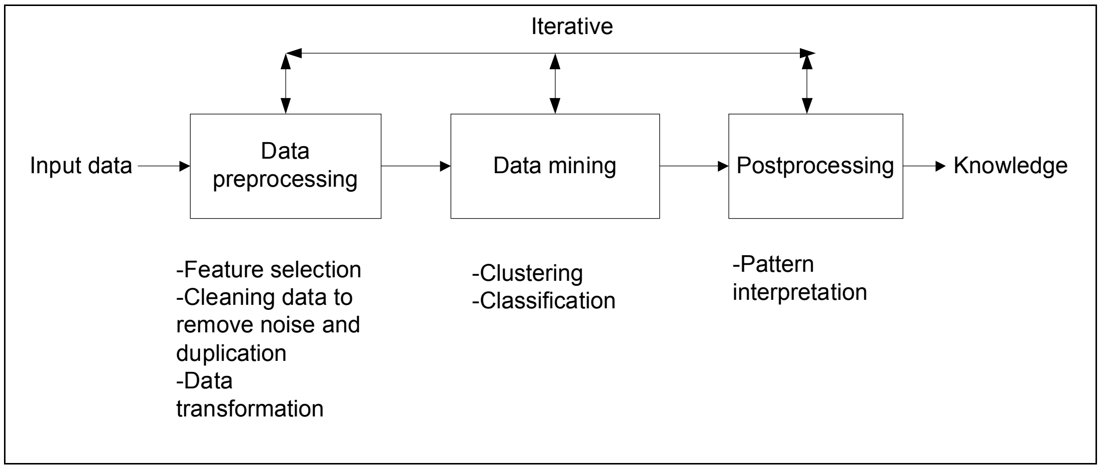


Figure 3.4. KDD Processes.

The data pre-processing function is intended to transform the worm’s raw data into an appropriate format for the next stage of the analysis, which is data extraction. The steps involved in this phase include feature selection, data cleansing to remove any noise, duplication or outlier and data transformation. The data pattern extraction is achieved using data mining; clustering and classification are two of the most common techniques used in data mining. The type of algorithm implemented under clustering (example: k-means) or classification (examples: Decision Tree, Support Vector Machine and Multilayer Perceptron) totally depends on the goal that is sought by the end of the KDD processes. Once the patterns are extracted from the data, they will be interpreted to ensure only valid and useful information or knowledge is kept for further exploration. All the KDD processes are iterative to ensure the result achieved is rigorous. Figure 3.4 displays common KDD processes involved in developing knowledge.

3.2.4 STAKCERT KDD Processes

Enhancements have been made to the KDD data pre-processing and pattern extraction process. Under the data pre-processing process, the static and dynamic analyses are implemented using the incident response standard operating procedures (SOP). While under the pattern extraction process, statistical methods comprising Chi-square and symmetric measure and security metrics are also introduced, as illustrated in Figure 3.5.

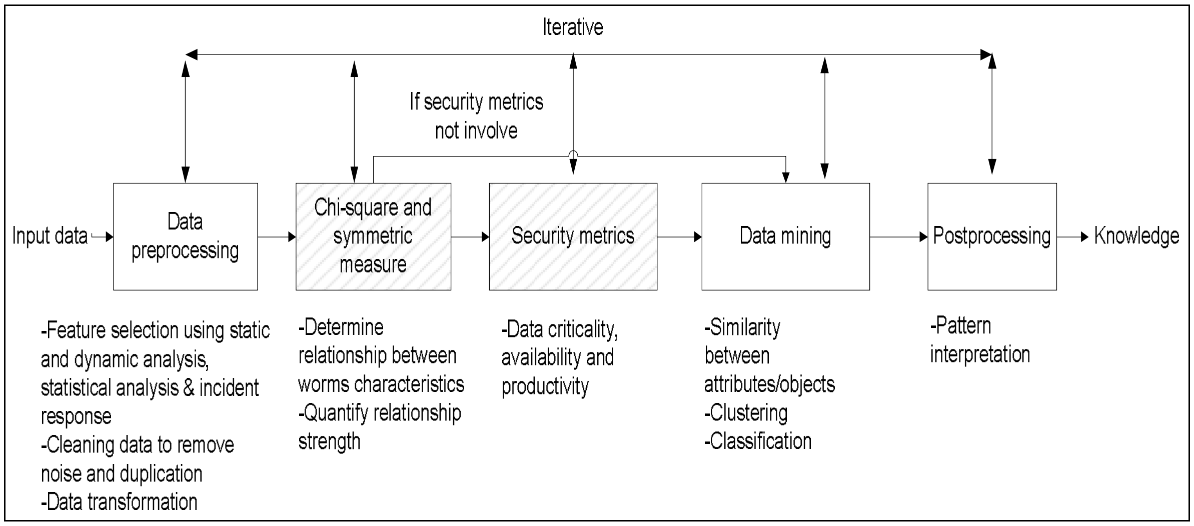

Iterative
If security metrics not involve
Input data
Data preprocessing
Chi-square and symmetric measure
Security metrics
Data mining
Postprocessing
Knowledge
-Feature selection using static and dynamic analysis, statistical analysis & incident response
-Cleaning data to remove noise and duplication
-Data transformation
-Determine relationship between worms characteristics
-Quantify relationship strength
-Data criticality, availability and productivity
-Similarity between attributes/objects
-Clustering
-Classification
-Pattern interpretation

Figure 3.5. STAKCERT KDD Processes.

3.2.4.1 Data Pre-processing

The raw worm and benign executables data received from the VX Heavens source needed to be transformed into a format that could easily be used for subsequent analysis. This is the stage at which feature selection, followed by cleansing data and data transformation, is carried out. When this research was conducted, the data pre-processing procedures accounted for almost 40% of the time taken for the whole research process. The following are details of each process involved during this phase:

I. Feature selection

In this research, the data from the VX Heavens source was retrieved in multiple formats. In order to use this data, it needs to be transformed into an understandable format; hence the need for feature selection using static and dynamic analyses. It should be remembered that feature selection is a search strategy process where only relevant data is chosen with the goal that the selected data can be valid and useful for the subsequent analysis. In this thesis, the chosen data, as already defined under section 3.2.1, was analysed using static and dynamic analyses in a controlled lab environment (refer to section 3.2.2).

Before and during the static and dynamic analyses, the incident response approach was applied. Standard operating procedures before and during the analysis must be followed and all the related procedures documented. Initially, all the listed software in Table 3.1 was checked to ensure all were already installed and working properly. Secondly, the condition of the testing computers and the network setting for each computer were checked. Thirdly, it was ensured that all the monitoring and test results were being documented. This was to make certain that there is always documentation if anything needs to be referred later. With reference to the incident response methodology by Prosise *et al.* (2003), as illustrated in Figure 3.6, in order to reach any solution which includes recovery steps or to implement security measures, all these seven steps play an important role.

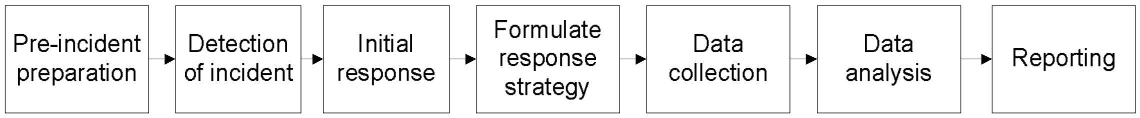


Figure 3.6. Incident Response Methodology.

However, according to the SANS Institute, six steps are required to handle any incident effectively, namely: preparation, identification, containment, eradication, recovery, and lessons learned (SANS 2008). Indeed, MyCERT used the SANS steps to produce the computer worm incident handling standard operating procedures (MyCERT 2002–see Figure 3.7). Therefore, in the STAKCERT KDD processes, the incident response methodology by Prosise *et al.* (2003) and SANS (2008), together with the MyCERT SOP for worm handling, are used as a basis and a guide.

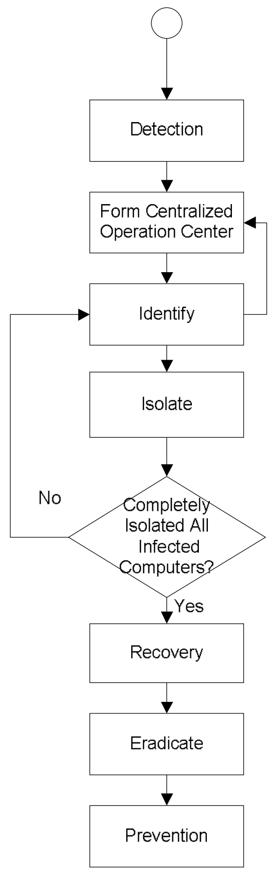

Detection
Form Centralized Operation Center
Identify
Isolate
No
Completely Isolated All Infected Computers?
Yes
Recovery
Eradicate
Prevention

Figure 3.7. MyCERT Worm IH SOP.
Adapted from MyCERT MA-041.052002:
Computer Worm Incident Handling Standard Operating Procedure 2002.

The incident response methodology and MyCERT SOP in worm handling are reflected in STAKCERT KDD Processes in step number 3 and number 5 (refer to Figures 3.1). These are designated as ‘Define the research design’ and ‘Integrate static, dynamic analysis and incident response procedures’ accordingly, where incident response is integrated. Before data analysis starts, preparation is carried out by examining the checklist to ensure all the installed software is working properly with the right testing lab network settings. Furthermore, all the analyses and findings are documented for all the experiments conducted. Table 3.2 displays how findings and analysis are documented.

Table 3.2. Documentation Template for Worm Analysis.

Activity	Observed Results
1. Load specimen into victim computer	
2. Run anti-virus program	
3. Analyse the anti-virus results and the file names	
4. Conduct strings analysis	
5. Look for scripts	
6. Conduct binary analysis	
7. Disassemble code	
8. Reverse-compile code	
9. Monitor file changes	
10. Monitor file integrity	
11. Monitor process activity	
12. Monitor local network activity	
13. Scan for open ports remotely	
14. Scan for vulnerabilities remotely	
15. Check promiscuous mode locally and remotely	
16. Sniff network activity	
17. Monitor registry activity	
18. Check registry changes	
19. Run code with debugger	

As a whole, in the STAKCERT KDD processes, the incident response is already integrated in worm detection, analysis and isolation. It is hard to separate incident response, since it plays an important role in the security field, especially in responding to worm incidents.

II. Static analysis

Static analysis is also known as white box analysis. It involves analysing and understanding source code where the worm code is not executed, which is opposed to dynamic analysis. Dynamic analysis involves executing the worm and watching its actions. The steps involved in static analysis are anti-virus checking, strings analysis, scripts analysis, binary analysis and disassembling; these stages are illustrated in Figure 3.8.

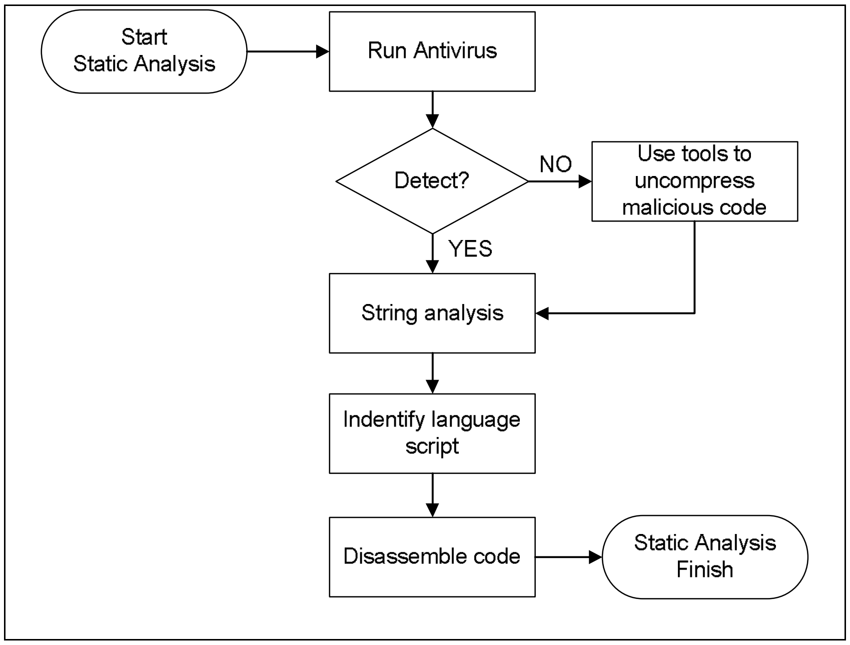


Figure 3.8. Static Analysis.

Static analysis is very effective in identifying the program flow, any files associated with the worm and any flaws or programming and implementation errors in the worm code, without actually running the worm code. In certain conditions, if only binary code is available, it has to be compiled to access the source code. This static analysis is in contrast with dynamic analysis which is explained in the next sub section.

A) Anti-virus checking

When the worm specimen has been copied to the test computer, the anti-virus program is run to check if it detects anything. If the anti-virus detects the worm, the worm’s name is identified and further information is accessible on any anti-virus website. The format of the specimen is also verified. If it is in a compressed or archived form, it will be decompressed or unpacked.

B) String analysis

An alternative way to identify the worm’s characteristics and functions is via extracting the strings from the worm specimen. Strings.exe is used to extract the strings; TDS-3 can also be used for string extraction. The information that could be retrieved from the extracted strings comprises the worm specimen’s name, user dialogue, password for backdoors, URLs associated with the malware, the email address of the attacker, help or command-line options, libraries, function calls and other executables used by the specimen.

C) Script analysis

The language written for the worm can be identified based on strings extracted from it. Table 3.3 can be used as guidance.

Table 3.3. Programming and Scripting Language.

Programming and Scripting Language	**Identifying Characteristics Inside the File**	**File's Common Suffix**
Perl	Start with line !#usr/bin/perl	.pl, .perl
Bourne Shell Scripting language	Starts with line !#/bin/sh	.sh
C	C programming language	.c
C++	Can be standalone program or many files referenced within the language	.cpp
Java	Contain java source code.	.java, .j, .jav
Assembly Language	Close to binary machine code	.asi
Active Server Page (ASP)	Can be built using Visual Basic, Jscript or Perl. Can combine HTML, scripts, Active-X server components.	.asp
JavaScript	Includes the word javascript or JavaScript, especially in the form <Script language = "JavaScript">	.js, .html, .htm
Visual Basic Script (VBScript)	Includes the word VBScript, or characters vb scattered throughout the file	.vbs, .html, .htm

D) Disassemble code

Disassemble and debugger codes are used to convert a raw binary executable into assembly language for further analysis. Ida Pro and OllyDbg are used to disassemble and debug the computer worm.

III. Dynamic analysis

Dynamic analysis involves executing the worm and watching its actions. The worm is activated in a controlled lab computer. The steps involved in dynamic analysis are: Monitoring file activities, monitoring processes, monitoring network activities and monitoring registry access. All of these are illustrated in Figure 3.9.

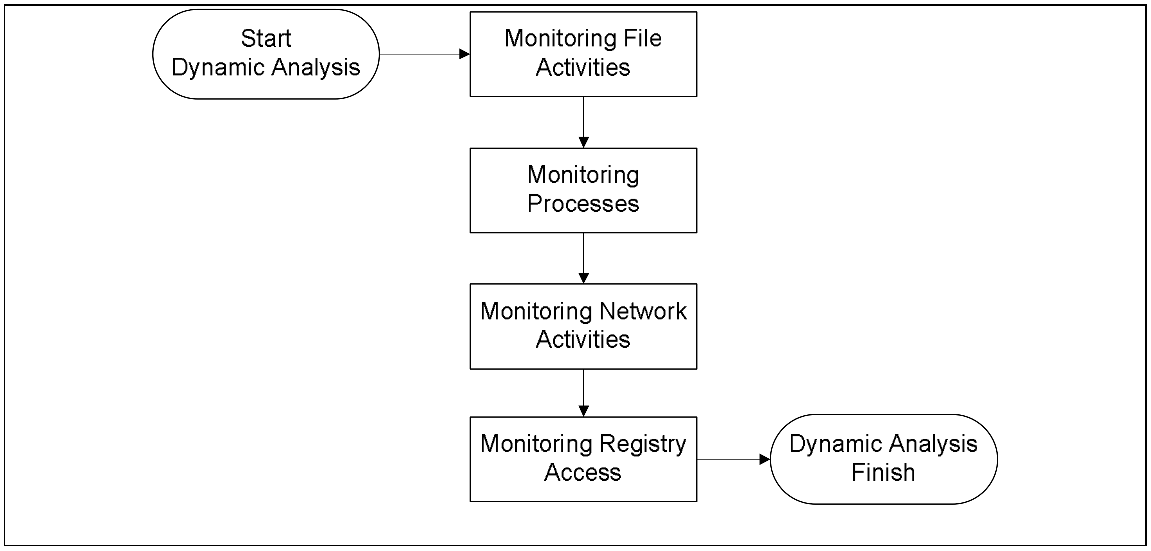

Start
Dynamic Analysis
Monitoring File Activities
Monitoring Processes
Monitoring Network Activities
Monitoring Registry Access
Dynamic Analysis Finish

Figure 3.9. Dynamic Analysis.

A) Monitoring file activities

Most computer worms read from or write to the file system. It might attempt to write files, alter existing programs, add new files or append itself to the file system. By using a tool such as Filemon, all actions associated with opening, reading, writing, closing and deleting files can be monitored.

B) Monitoring process

A monitoring tool such as Prcview v3.7.3.1 or Process Explorer displays each running program on a computer, showing the details of what each process is doing. With this tool, the files, registry keys and all of the DLLs that each process has loaded can easily be monitored. For each running process, the tool displays its owner, its individual privileges, its priority and its environment variables.

C) Monitoring network activities

From a remote computer, which will be in the same LAN as the infected testing computer, the port scanner, Nmap program and a sniffer will be installed.

The port scanner and Nmap program are used to monitor the listening port. A sniffer will be installed to sniff the worm traffic. All of the related tools like Ethereal, NeWT and TDS-3 use the sniffer. By using the sniffer, details of individual packets and all packets transmitted across the LAN can be monitored. In addition, the local network monitoring tool (TDIMon) will monitor and record all requests to use the network interface and show how the worm grabbed the network resources and used them.

The worm might have placed the network interface in promiscuous (broadcast) mode, which allows it to sniff all packets from a LAN. To determine if the infected computer is in the promiscuous mode state of interface, the Promiscdetect.exe tool must be run.

D) Monitoring registry access

The registry needs to be monitored, as it is the hierarchical database containing the configuration of the operating system and most programs installed on the computer. The monitoring of registry access is carried out by using Regmon.

IV. Data cleaning and transformation

The data cleaning process that is part of the data pre-processing process is already conducted under the data source section. This is where all the duplicates, noise and outlier data are removed. When conducting the static and dynamic analysis, a pattern of worm characteristics is identified. Each dataset has its own way of being recognised and simplified. This leads to the selection of useful worm characteristics, which are: The worm payload, infection, propagation, operating algorithm and activation. Selection of the wrong worm characteristics (also known as attribute selection) might lead to inaccurate results and wasted time. Later, these five worm characteristics are used to represent all the datasets used for the experiments. Then, to use these datasets in SPSS and data mining (using JAVA–WEKA), the worm characteristics are transformed into nominal data with a certain number representation.

Furthermore, in this research, the dataset from the VX Heavens source consists of executables source code in the Windows PE format (i.e. file name executables .cpl, .exe, .dll, .ocx, .sys, .scr, and .drv) and some of them in programming and scripting language (i.e. .pl, .sh, .c, .cpp, .java, .vbs). If the source code was not executable, the static analysis was conducted to extract the main features of the worm, which later are transformed into an understandable format as an input for WEKA software. As for the executable source code, the dynamic analysis was conducted. In certain condition, both static and dynamic analyses were conducted to extract the main patterns or features of the worm, which subsequently were used as input for machine learning algorithm (WEKA software).

From the worm source code, once it has been analysed using the static or dynamic analysis, the five main features of the worm algorithm are being extracted into semi format structure comprising five different of subareas which are the payload, infection, activation, operating algorithm and propagation to capture the worm characteristics. These five different subareas (refer to Table 3.4) later is transformed into nominal data with five numeric values which are used as the input to the machine learning algorithms, where the WEKA software is used.

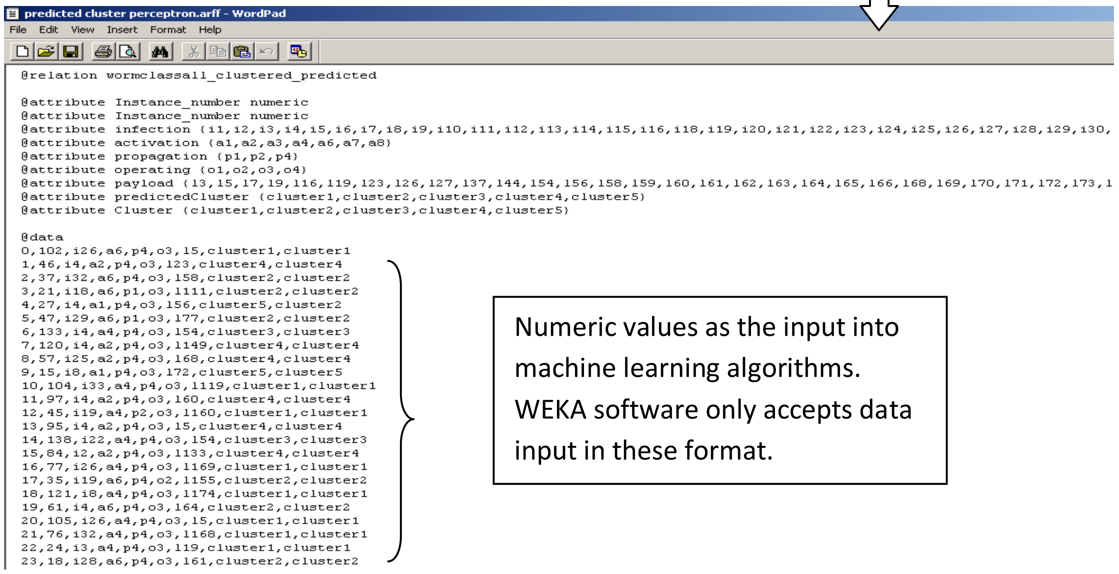
Table 3.4. Example of Data Transformation.

Dataset 1					**New format for dataset 1**
Infection	**Activation**	**Propagation**	**Operating algorithm**	**Payload**	
File, email and sharing directories	Self activation	Random	Terminate stay resident	Backdoor and autorun registry	*i21,a4,p1,o3,l59*

worm characteristics are extracted from the worm source code

transformed worm code into nominal data with numeric values where:

i21 represents infection – as file, email and sharing directories,
a4 represents activation – as self activation,
p1 represents propagation – as random,
o3 represents operating algorithm – as terminate and stay resident
and *l59* represents payload – as backdoor and autorun registry.


predicted cluster perceptron.arff - WordPad
File Edit View Insert Format Help

```
@relation wormclassall_clustered_predicted

@attribute Instance_number numeric
@attribute Instance_number numeric
@attribute infection {i1,i2,i3,i4,i5,i6,i7,i8,i9,i10,i11,i12,i13,i14,i15,i16,i18,i19,i20,i21,i22,i23,i24,i25,i26,i27,i28,i29,i30,
@attribute activation {a1,a2,a3,a4,a6,a7,a8}
@attribute propagation {p1,p2,p4}
@attribute operating {o1,o2,o3,o4}
@attribute payload {l3,l5,l7,l9,l16,l19,l23,l26,l27,l37,l44,l54,l56,l58,l59,l60,l61,l62,l63,l64,l65,l66,l68,l69,l70,l71,l72,l73,l
@attribute predictedCluster {cluster1,cluster2,cluster3,cluster4,cluster5}
@attribute Cluster {cluster1,cluster2,cluster3,cluster4,cluster5}

@data
0,102,i26,a6,p4,o3,l5,cluster1,cluster1
1,46,i4,a2,p4,o3,l23,cluster4,cluster4
2,37,i32,a6,p4,o3,l58,cluster2,cluster2
3,21,i18,a6,p1,o3,l111,cluster2,cluster2
4,27,i4,a1,p4,o3,l56,cluster5,cluster2
5,47,i29,a6,p1,o3,l77,cluster2,cluster2
6,133,i4,a4,p4,o3,l54,cluster3,cluster3
7,120,i4,a2,p4,o3,l149,cluster4,cluster4
8,57,i25,a2,p4,o3,l68,cluster4,cluster4
9,15,i8,a1,p4,o3,l72,cluster5,cluster5
10,104,i33,a4,p4,o3,l119,cluster1,cluster1
11,97,i4,a2,p4,o3,l60,cluster4,cluster4
12,45,i19,a4,p2,o3,l160,cluster1,cluster1
13,95,i4,a2,p4,o3,l5,cluster4,cluster4
14,138,i22,a4,p4,o3,l54,cluster3,cluster3
15,84,i2,a2,p4,o3,l133,cluster4,cluster4
16,77,i26,a4,p4,o3,l169,cluster1,cluster1
17,35,i19,a6,p4,o2,l155,cluster2,cluster2
18,121,i8,a4,p4,o3,l174,cluster1,cluster1
19,61,i4,a6,p4,o3,l64,cluster2,cluster2
20,105,i26,a4,p4,o3,l5,cluster1,cluster1
21,76,i32,a4,p4,o3,l168,cluster1,cluster1
22,24,i3,a4,p4,o3,l19,cluster1,cluster1
23,18,i28,a6,p4,o3,l61,cluster2,cluster2
```

Numeric values as the input into machine learning algorithms. WEKA software only accepts data input in these format.

The formation of the new STAKCERT worm classification and STAKCERT worm relational model, which are the subsequent processes after the static and dynamic analyses, are not explained in this chapter. The details can be found in Chapter 4 (section 4.3).

3.2.4.2 Chi-square and Symmetric Measures

Once the data pre-processing process is completed, statistical analysis is conducted to analyse the datasets. The statistical analysis gives added value to data mining analysis (Giudici 2010). For this research, the Chi-square, symmetric measure, Euclidean distance and 10-cross validation under data mining are implemented. Details of Euclidean distance and 10-cross validation are explained under data mining in section 3.2.4.4.

To test the relevance of the STAKCERT worm classification and the STAKCERT relational model, Chi-square and symmetric measure tests are used. These tests are used to determine the relationship which exists between worm characteristics chosen in the STAKCERT relational model, followed by the symmetric measure to quantify the strength of the relationship.

Chi-Square is a statistical test for cross tabulation which works by comparing the result of the actual frequencies and the expected frequencies to verify whether the result happens by chance or not (Greasley 2008). Indeed, it is also capable of measuring the discrepancy between the observed cell counts (from the experiment) and what would be expected if the rows and columns were unrelated. The Chi-square formula used on these data is displayed in equation

1, where *O* stands for observed frequency, *E* stands for expected frequency and X^2 for Chi-square.

$$X^2 = \frac{(O-E)^2}{E} \tag{1}$$

Expected frequencies are those which would be expected if data were randomly distributed. The expected count in this cell is the average count which would be anticipated under the null hypothesis. In general, the expected count for each cell of the contingency table is calculated as displayed in equation 2.

$$Expected\ Count = \frac{RowTotal * ColumnTotal}{GrandTotal} \tag{2}$$

The Chi-square test becomes invalid if the expected frequency is less than 5. Since the dataset is categorical (also known as nominal) data, testing was conducted based on the frequencies. They are later converted into percentage format for further analysis. Software SPSS has been used to conduct this statistical analysis.

The Chi-square and symmetric measures involve null hypothesis (H_0) and alternative hypothesis (H_a). The null hypothesis (H_0) states that there is no significant difference between expected and observed frequencies. In other words, there is no relationship between features. If there were no relationship between the features, the observed and the expected count would be similar (equal to 0). The alternative hypothesis (H_a) states that they are different. Thus, if $H_{0,}$ is rejected, it can then be concluded that there is a relationship between the features. The level of significance chosen is 95% confidence; in other words, the benchmark where the difference is not due to chance alone is set at 0.05. If the significance or probability (p) value is less than 0.05, it means there is a less than 5 out of 100 probability that it happened by chance. Details of the Chi-square and symmetric measure and how they are applied can be found in Chapter 4 (section 4.4.2).

If the expected counts for the nominal data are less than five, with the condition that it is a 2x2 contingency table (the number of degrees of freedom is always 1), the alternative test that can be carried out is known as Fisher's exact test (Weisstein 2011). Fisher's exact test formula is displayed in equation 3.

$$Y = \frac{([O-E] - 0.5)^2}{E} \tag{3}$$

where,

Y= Fisher's Exact Test

O= Observed frequency

E= Expected frequency

This Fisher's exact test has the same objective as the Chi-square test, but it is dedicated to expected counts of less than five. As discussed, in Chi-square the expected counts should be more than five. This is the adjusted formula where only one side is being used, which results in the two-sided significance value being halved. Hence, the value of exact significance 1 sided is considered to be the result. Details of how Fisher's exact test is applied can be found in Chapter 4 (section 4.4.2).

Since the data involved in this research is nominal data, it therefore can be summarised that the importance of applying the Chi-square and symmetric measure in this research is due to its functionality, which enables the determination of the relationship existing between worm characteristics and the strength of the relationship chosen in the STAKCERT relational model to be quantified.

3.2.4.3 Security Metrics Method

Two important attributes being measured when conducting a depth study in this research are weight and severity. In order to decide how to assign the weight and severity values, which are explained in detail in Chapter 5, a solution known as security metrics is used. Security metrics is a method that helps to quantify, classify and measure information on security operations. In security metrics, the studied threats are defined, then threats are transformed into metrics or representations that can easily be measured. Then understand and identify the vulnerabilities, flaws, problems, weaknesses or damage they can cause to the security infrastructure, check the existing countermeasure process performance and, if necessary, recommend the improvement of any technology or countermeasure process (Jaquith 2007).

The security metrics processes are already being applied in STAKCERT KDD Processes for worm detection and worm response as displayed in Table 3.5.

Table 3.5. Security Metrics in STAKCERT Processes.

Security metrics processes	**Applying security metrics in STAKCERT**
1) Define worm threats	Yes
2) Represents worm threats into metrics	Yes. • Worm data is represented based on payload, infection, activation, propagation and operating algorithm. • Formation of the STAKCERT worm classification and STAKCERT relational model.
3) Understand and identify the vulnerability, flaw, problem, weakness and damage to security infrastructure	Yes. • Run the static and dynamic analysis. • Identify the need to assign weight and severity value to assign the countermeasure process.
4) Check the performance of the existing countermeasures	Yes. • Integrate and run data mining using JAVA-WEKA to check the accuracy rate of weight and severity assigned.
5) Recommend any technology or countermeasure process for improvement	Yes. • Apoptosis to isolate the most severe worm attacks.

For STAKCERT research, in order to understand the threat posed by a worm, a deep and thorough understanding of worm architecture is necessary; in this thesis, this led to the formation of STAKCERT worm classification and the STAKCERT worm relational model. Initially, the characteristics that need to be observed are defined. Then, during the static and dynamic analysis, the worms are analysed and simplified into worm representation, which comprises payload, activation, operating algorithm, infection and propagation.

A thorough analysis related to the vulnerabilities, flaws, problems, weaknesses or the damage the worm can cause to the security infrastructure is closely monitored. As a result, weight and severity are chosen as two main attributes in assigning the countermeasure process. Detailed reasons for the selection of weight and severity can be found in Chapter 2, section 2.2.2.

To analyse the performance of a worm that has already been assigned with different weight and severity values, it is tested using the JAVA-WEKA software, in which different data mining algorithms are also integrated. As a result, all worms with a high severity level are recommended to be isolated using the apoptosis concept.

Apart from the elements stated above, security metrics can also be measured based on the perimeter defence, control and coverage, availability and reliability and application risks. All these measurements were already taken into consideration when the worm analysis was conducted. Therefore, as a result, the weight and severity performance and value are tested based on data criticality level, infrastructure availability and loss of productivity. Moreover, an algorithm has been developed using the above as a basis. Details of this algorithm can be found in Chapter 5 (section 5.3.1).

Lastly, the main reason why security metrics method has been chosen in this research is due to its capabilities to make the job of defining, understanding, identifying and measuring information security efficient, accurate, measurable and reliable. This is also supported by Atzeni and Lioy (2006), where they state that work can be more profitable if it is enhanced using the security metrics and is more efficient if it is measurable.

3.2.4.4 Data Mining

Clustering and classification play important roles in data mining. Clustering is also known as unsupervised learning, while classification is known as supervised learning. Both of these techniques have been applied in this research. However, it must be remembered that the datasets used in this research are nominal data.

I. Clustering

Earlier, the STAKCERT worm relational model was tested using the Chi-square and symmetric measures. Subsequently, the k-means clustering technique is used to cluster all the datasets into different types of worm group or type. For STAKCERT research, five different types of worm group have been identified, further details of which can be found in Chapter 4 (section 4.4.3.1). In k-means, datasets are partitioned based on centroids, also known as mean. The basic steps of how the k-means works are as follows: firstly, the number of clusters is chosen. Secondly, the datasets are assigned to their closest cluster centre based on Euclidean distance(ED). The Euclidean distance equation is displayed in equation 4 where *x* and *y* are two different objects and the Euclidean distance is the square root of the summation from the squares of the differences between *x* and *y* values.

$$ED = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \qquad (4)$$

Thirdly, the centroid of each cluster is calculated and taken as the new centre value for each of the clusters. Lastly, the whole process is repeated with a new cluster centre until the same point is assigned to each cluster. The k-means is chosen due to its effectiveness and simple method. WEKA is used to apply the k-means technique. WEKA is open source software, implemented in JAVA and it has a collection of machine learning algorithms to solve data mining problems (Hall et al 2009).

II. Classification

Referring to Figure 3.1, processes 8 and 12 involve data mining. With regard to process number 8, once the clustering is complete, the clusters of the five different types of worm are integrated with five different classification algorithms. Earlier on, the clustering is meant to obtain the label or the five different groups of worm type. In order to test the accuracy of the five different types of worm assignment, the classification algorithms are integrated (the findings can be examined in Chapter 4, section 4.4). The classification algorithms chosen are the Sequential Minimal Optimization (SMO), Multilayer Perceptron (MLP), Naïve Bayes, Decision Tree (J48) and K-nearest Neighbours (IBk). Details of the above classification algorithms can be found in Table 3.6.

Table 3.6. Classification of Algorithm Functions.

Name	**Function**
Naïve Bayes	Standard probabilistic Naïve Bayes classifier where it used as an estimator and probability technique.
J48	To generate a pruned or unpruned C4.5 Decision Tree. It is the descendent of ID3.
Multilayer Perceptron	To train and test data using backpropagation in a neural network
SMO	To train and test data using the sequential minimal optimisation algorithm for support vector classification.
IBk	It is the k-nearest neighbour classifier

These classifications are applied so that a comparison can be made between these algorithms, which therefore enable identification of the most accurate classification algorithm. This concept is applied once again in process number 12 (also from Figure 3.1) to different attributes which are weight and severity (details of findings can be found in Chapter 5, section 5.3.2). While in Table 3.7, are the configuration settings used for the testing conducted.

Table 3.7. WEKA Classification Algorithms Configuration.

Algorithm name	**Configuration**	**Description**
Naïve Bayes	weka.classifiers.bayes. NaiveBayes	False for debug, display mode in old format, kernel estimator and supervised discretization.
J48	weka.classifiers.trees.J48 -R –N 7 –Q 3 -M 2	Binary splits: false, reduced error pruning with confidence of factor for pruning= 0.1, number folds=7, seeds for randomizing the data=3 and restrict the minimum number of instances in a leaf=2.
Multilayer Perceptron	weka.classifiers.functions. MultilayerPerceptron —L 0.3 –M 0.2 –N 500 –V 0 –S 0 –E 20 –H 0	The learning rate= 0.3, momentum = 0.2, training time= 500, validation set size=0, seed=0, validation threshold = 20 and hidden layer =0.
SMO	weka.classifiers.functions.SMO -C 1.0 -L 0.0010 -P 1.0E-12 -N 0 -V -1 -W 1 -K “weka.classifiers.functions.supportVector.PolyKernel -C 250007 -E 1.0”	Build logistic models=false, complexity parameter=1.0, checks turned off=false, debug=false, epsilon for round-off error=1.0E-12, data transformation = normalize training data, kernel=polykernel with cache size 250007 and exponent 1.0, number folds=use training data, random number seed for the cross validation=1 and tolerance parameter=0.0010.
IBk	weka.classifiers.lazy.IBk -K 8 -W 0 -I -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\"”	The number of neighbours to use=8, cross validate=false, debug=false, the distance weighting method used=weight by 1/distance, mean squared =false, the nearest neighbour search algorithm to use=Euclidean distance and window size =0, where no limit to the number of training instances.

III. STAKCERT Worm Apoptosis Algorithm

Once the security metrics processes are complete, a set of STAKCERT rules are formed based on the implications from the data criticality level, infrastructure availability and loss of productivity perspectives. These rules are part of the STAKCERT worm apoptosis algorithm and presented in IF-THEN-ELSE form. Basically, the rules are expressed in the form of:

If (Attribute-1, value -1) and (attribute -2, value -2) and….

and (attribute –n, value –n) then (decision, value)

The decision made is the dependent variable, since it relies on the worm's selected attributes, which are: The payload, activation, infection, operating algorithm and propagation. Each of these attributes are assigned with a weight that is either low, medium or high, based on the worm implications (using security metrics method, refer Table 3.5). The next decision is the severity level, which leads to the apoptosis condition. The severity level is categorised as low, medium or high. Details of the weight assignment, severity value categorisation and the rules and algorithm can be found in Chapter 5. Once the datasets have been assigned with the related weight and severity values, the performance criteria of the STAKCERT worm apoptosis algorithm is verified based on the accuracy and false positive rate.

IV. Performance Criteria Definition

The performance criteria is also applied to the whole STAKCERT model. The accuracy also refers as the correct classification. The *false positive* (*FP*) means the data is being misclassified as class A but actually it belongs to a different class and *false negative* (*FN*) occurs when the data is wrongly classified as a different class but actually it belongs to class A. While *true positive* (*TP*) occurs when data is correctly classified as class A and *true negative* (*TN*) occurs when data is correctly classified wrong in class A. So the correct classifications are the *TP* and *TN*. The *FP rate (FPR)* is the *false positive (FP)* divided by the summation of *false positive (FP)* and *true negative (TN)*. While the *TP rate* (*TPR*) is *true positive* (*TP*) divided by the summation of *true positive* (*TP*) and *false negative* (*FN*). *Precision* is the proportion of relevant documents in the results returned and *Recall* is the ratio of relevant documents found in the search result to the total of all relevant documents (same like *TP* rate equation). The higher the *Precision* and *Recall* values mean the more relevant documents are returned more quickly. Lastly the *F-measure* is a way of combining *Recall* and *Precision* scores into a single measure of performance (Tewolde 2011). The equations used were the following:

$$\text{True positive rate} = TP / (TP + FN) \tag{5}$$

$$\text{False positive rate} = FP / (FP + TN) \tag{6}$$

$$\text{F-measure} = 2 * \text{recall} * \text{precision} / (\text{recall} + \text{precision}) \tag{7}$$

$$\text{Accuracy} = (TP + TN) / (TP + TN + FP + FN) \tag{8}$$

$$\text{Error rate} = 1 - \text{Accuracy} \tag{9}$$

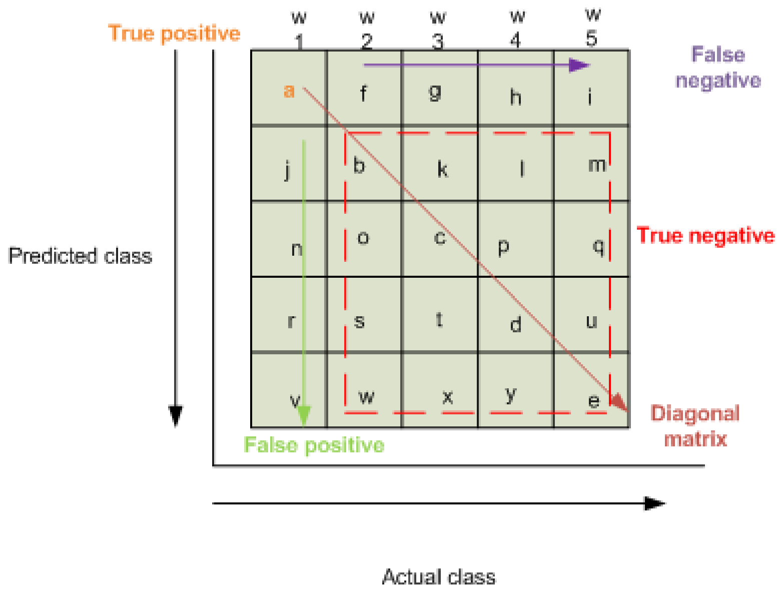
Precision = TP / (TP+ FP) *(10)*

False negative rate = FN / (FN + TP) *(11)*

A confusion matrix also known as a contingency table, is an easy way to describe experimental results. It is a matrix to show predictions and actual classifications (Kohavi and Provost 1998). The dimension of the confusion matrix is *m x m* where *m* is the number of different label values. An example of the confusion matrix and how different values are calculated is displayed in Table 3.8. For confusion matrix 5 X 5, class *w1* is used for an example where different colours are used to represent the terms.

Table 3.8. Examples of Confusion Matrix 2x2 and 5x5.

		Predicted class	
		Yes	No
Actual class	Yes	*true positive (TP)*	*false negative(FN)*
	No	*false positive (FP)*	*true negative(TN)*



Apart from the confusion matrix, a Receiver-Operating characteristics (ROC) curve is the alternative way to examine the classifiers performance (Swets 1988). It is useful in assessing the accuracy of the predictions. It is a graph plot with *X* axis representing the *FP rate* (*FPR*) and *Y* axis representing as the *TP rate* (*TPR*). Moreover, according to Scharenbroich (2003), the ROC curve identifies how many false positives acceptable to be guaranteed a certain percentage of true positives. An ideal ROC curve is the step-function. While ROC area represents the area under the ROC curve. For the ROC area, if the point is (0,1) it means all positive cases and negative cases are correctly classified. This indicates as the perfect classifier since the *FPR* is 0 (none) and the *TPR* is 1(all). If the point is (1,0) it shows the classifier is wrongly classified all the cases since the *FPR* is 1 and *TPR* is 0. While the point is (0,0) predicts all cases to be negative and when the point is(1,1) as all cases to be positive. An example of a ROC curve diagram can be seen in Figure 3.10. The *X* axis represents the *FP rate* and *Y* axis represents the *TP rate*.

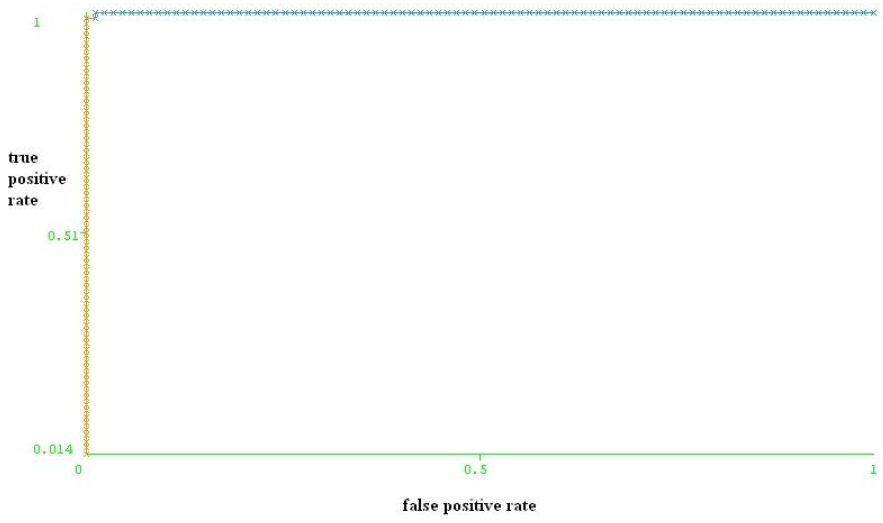


Figure 3.10. A ROC Curve Diagram

For evaluation, the 10-fold cross validation, which is also known as leave-one-out, is used to conduct this testing. This cross validation is widely used as a standard way of verifying the rule sets (Grzymala-Busse 2010). All cases are randomly reordered and then a set of all cases is divided into ten equal sizes. During each run, one of the partitions is used as the test and the rest are used for training. This process is repeated ten times so that each partition is used for testing exactly once. The reasons behind the choice of this kind of test are, firstly, that it uses as much data as possible for training and testing and secondly, the better accuracy of its findings. Details of the STAKCERT worm apoptosis algorithm and test results for the accuracy and *FP rate* can be found in Chapter 5.

3.2.4.5 Data Post-processing

At this stage, the complete pattern extracted from the data is interpreted so useful knowledge is produced by the end of all the processes. Later, the pattern extracted can be simplified using graphs or any suitable methods to represent the complete extracted pattern for any further exploration or analysis. In STAKCERT, at this point, a conclusion and summary can be made based on all the findings to ensure all the objectives for this research are achieved successfully.

3.3 Summary

In this chapter, the STAKCERT research processes used for this study are discussed. It is believed that these processes act as the backbone that provides guidance on how research should be developed and proper activities carried out. It ensures that a consistent and reproducible approach is used from the first activity of the research processes until all the processes are complete. Details of how these research processes are applied and the findings can be found in Chapter 4 and Chapter 5.

CHAPTER 4

MODELLING STAKCERT FOR WORM DETECTION

Chapter 4 outlines the STAKCERT model for the detection of worm infection. This chapter discusses the methods integrated within the STAKCERT model, in terms of worm detection, and a new STAKCERT worm classification and relational model are introduced in this research as part of the STAKCERT model. The experimental results gained from the use of the STAKCERT model were compared with existing works and it was found that the STAKCERT model successfully addressed the problems left by existing works: STAKCERT yielded a 98.75% accuracy rate for worm detection, using the Multilayer Perceptron algorithm.

4.1 Introduction

Chapter 3 discussed the enhancements of the data-preprocessing processes, which consist of the added processes of the Chi-square, symmetric measures and security metrics. In this chapter, all the processes involved in the STAKCERT KDD processes (Figure 3.5 in Chapter 3) are applied within the STAKCERT model for worm detection are explained. Generally, there are 2 phases involved in the formation of the STAKCERT model, which are worm detection (Phase 1) and worm isolation (Phase 2) (as displayed in Figure 4.1). In this chapter, this thesis focuses on phase 1. As displayed in Figure 4.1, there are five main processes involved in phase 1, which are worm detection, worm analysis, STAKCERT worm classification and the data matching processes. Each of these processes plays an important role and has its own integrated processes, which are explained in detail in Section 4.3.

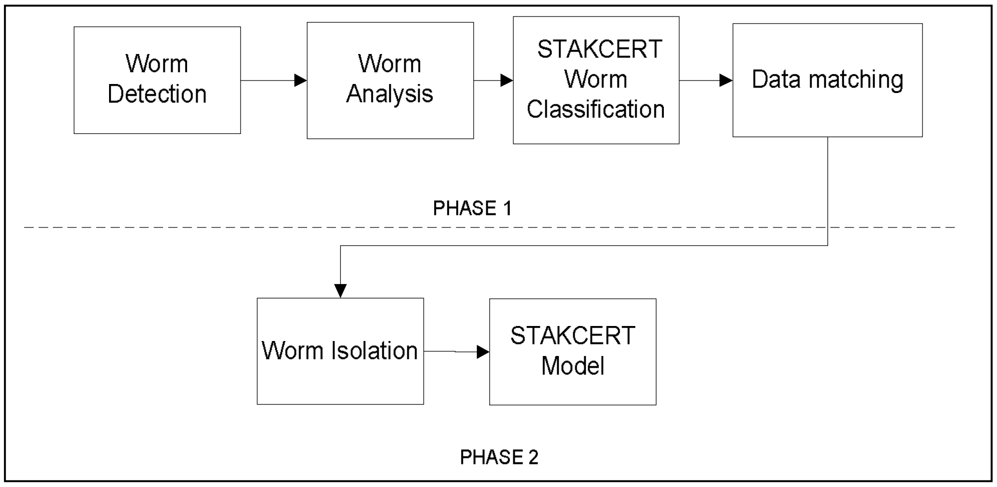


Figure 4.1. An Overview of STAKCERT Phases 1 and 2.

4.2 Related Works

Prior to the introduction of the STAKCERT model of worm detection, a thorough study of the existing literature on worm architecture, worm implication, worm detection and worm response issues was undertaken. Such literature was reviewed in order to see where further improvements could be made and it was ascertained that the studies on worm architecture and the threat implications should be the initial consideration in producing improved worm detection and better response techniques. These details have already been discussed and outlined in Chapter 2. Furthermore, in Chapter 2 a few works on worm detection using different methods and algorithms are discussed.

In order to test the effectiveness of the STAKCERT model for worm detection, a comparison of the work with research conducted by Siddiqui *et al.* (2009) and Dai *et al.* (2009) was undertaken. Both works used the same datasets as in this thesis and had the same objective; i.e., to detect worms and increase the worm detection rate. Indeed, Siddiqui *et al.* (2009) used the static features of a worm programme, while Dai *et al.* (2009) incorporated dynamic instruction sequence mining techniques involving the runtime features of a worm programme. These two works are the closest to this thesis and this thesis has focused on bridging the gaps that arose from the aforementioned works by integrating static features and dynamic analysis within the STAKCERT model. In terms of performance, Siddiqui *et al.* (2009) yielded a better accuracy rate of 96% by using random forest, while Dai *et al.* (2009) detection rate was 91.9% by using SVM. Their results and the results of this thesis are further discussed in Section 4.4 and it became apparent that this thesis has outperformed both these accuracies, with a 98.75% success rate (using the Multilayer Perceptron). Improvement in the STAKCERT KDD processes, the integration of STAKCERT worm classification and STAKCERT relational model and performance optimisation by using MLP algorithm, were the key factors in this achievement, as explained in detail in the next section.

4.3 STAKCERT Model for Worm Detection

Phase 1 of worm detection is outlined in detail in Figure 4.2 below.

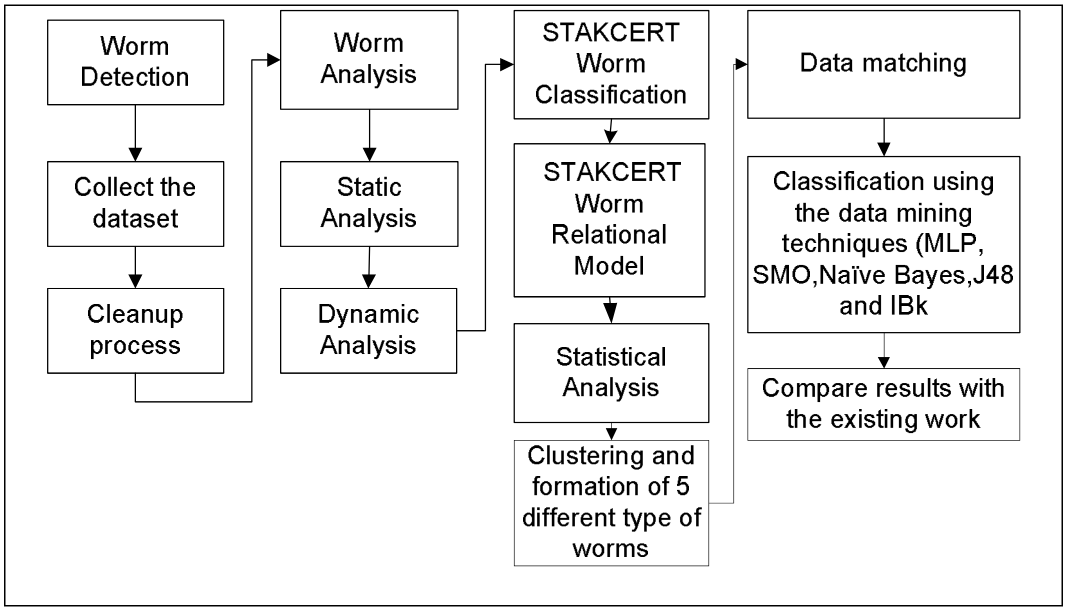

Worm Detection
Collect the dataset
Cleanup process
Worm Analysis
Static Analysis
Dynamic Analysis
STAKCERT Worm Classification
STAKCERT Worm Relational Model
Statistical Analysis
Clustering and formation of 5 different type of worms
Data matching
Classification using the data mining techniques (MLP, SMO,Naïve Bayes,J48 and IBk
Compare results with the existing work

Figure 4.2. Phase 1 of STAKCERT.

In terms of STAKCERT KDD processes, specifically data pre-processing and dataset collection, the cleanup processes and the static and dynamic analyses have already been ascertained. The worm detection and analysis were discussed in detail in Chapter 3, Section 3.2.4, while the STAKCERT worm classification and the involved data matching processes are explained in the next section.

4.3.1 STAKCERT Worm Classification

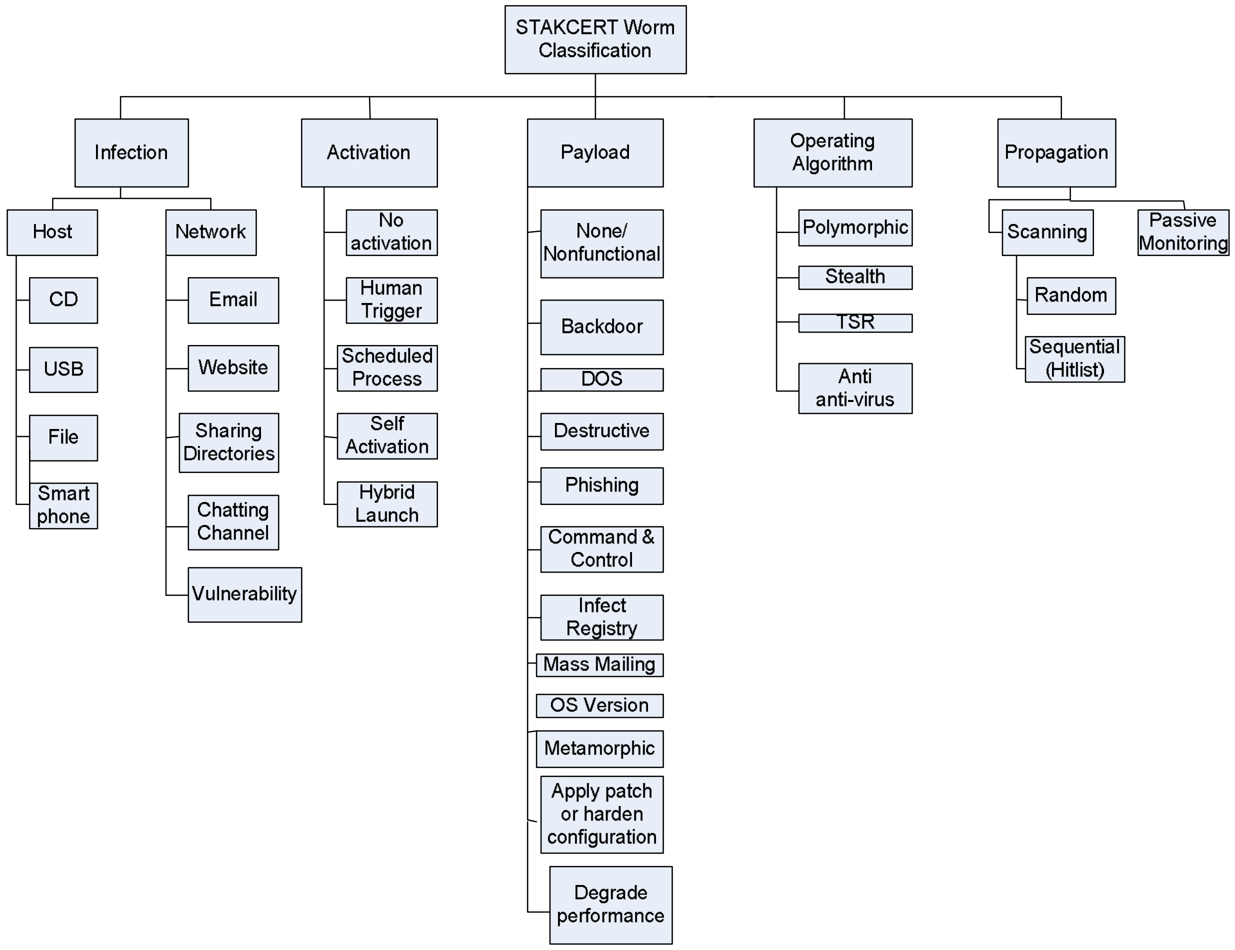


Figure 4.3. STAKCERT Worm Classification.

STAKCERT worm classification consists of five main attributes, which are infection, activation, payload, operating algorithm and propagation.

A) Infection

This is the phase concerned with how a computer becomes infected by a worm. There are two ways in which a worm infects a computer and these are via a host or a network. The host is a mechanism that the worm requires in order to copy itself to a new system that is not yet infected; a worm cannot autonomously propagate across a network. The host computer worm refers to where the original worm terminates itself after launching a copy onto another host. Thus, there is only one copy of the worm running elsewhere on the network at any given moment and human help is required in moving the worm from one computer to another. CD, USB (thumb-drive and external hard disk), file and smart phone are the most common hosts available today.

Whilst a network comprises multiple parts, each worm can run on different computers and perform different actions for communication purposes. Most worms simply copy themselves to a vulnerable computer that can share data, while most Windows networks allow computers within defined subgroups to exchange data freely, making it easier for a worm to propagate itself.

B) Activation

Activation is defined as a worm’s trigger mechanism and this phase refers to the worm entering the host, once it finds a computer.

I. No Activation

A worm with no activation will just remain within a computer, doing nothing other than taking up some hard disk space.

II. Human Trigger

The human trigger is the slowest activation mechanism, where email is commonly used as the medium with which to spread a worm. Then, social engineering techniques are used to encourage a user to click on the file and activate the worm Zou *et al.* (2004). According to Christoffersen and Mauland

(2006), some worms are activated when the user performs a certain activity, such as resetting the computer or logging onto the system, thereby running the login scripts or executing a remote infected file.

III. Scheduled Process

According to Weaver *et al.* (2003), the second fastest method of worm activation is through the use of scheduled system processes. A schedule process is an activation that is based on a specific time and date and many computer operating systems and applications include auto-update programmes; i.e., they periodically download, install and run software updates.

IV. Self Activation

The quickest way in which worms are activated is through the exploiting of vulnerabilities in services that are always on and always available (e.g., Code Red (Berghel 2001) exploiting IIS Web servers) or within the libraries that the services use (e.g. XDR (CERT 2002)). These worms either attach themselves to running services or execute other commands, using the permissions associated with the attacked service.

V. Hybrid Launch

The hybrid launch employs a combination of two or more activation mechanisms in order to launch a worm, with ExploreZip (Nanchenberg 1999) being an example of a hybrid-launch worm. Such a worm sends an e-mail that requires the user to launch the infected attachment, so that control of the system may be gained. Once activated, the worm automatically spreads itself to other computers over the peer-to-peer network. These targeted computers then become infected on the next reboot, without the requirement of user intervention. Stuxnet worm is another example of a hybrid-launch worm. It spreads itself by exploiting five Windows vulnerabilities and via network shares with weak passwords (Shearer 2010).

C) Payload

A payload is defined as the destructive mechanism of a worm and is a code designed to do more than spread a worm (Castaneda *et al.* 2004). Many worms have been created that are simply designed to spread without actually attempting to alter the systems they pass through.

I. No Payload

A worm with no payload does not do any harm to a computer system. Indeed, this kind of worm will just propagate without initiating any destructive mechanisms within a computer.

II. Installing a Backdoor

Backdoor is a term used to describe a secret or undocumented means of getting into a computer system. Many worms' programmes have backdoors incorporated into them by the worms' writers, so that they may gain access, in terms of troubleshooting or changing the programme. They create backdoors once they gain access, in order to allow themselves an easier way in, or in case their original entrance is discovered. An example of the worm is the Blaster worm (Bailey *et al.* 2005), which used the backdoor mechanism to transfer the worm payload to newly-infected systems.

III. Denial of Services

A denial of service (DoS) attack floods a network with an overwhelming amount of traffic, slowing its response time for legitimate traffic or grinding it to a halt completely. The more common attacks use the built-in features of the TCP/IP protocol, in order to create exponential amounts of network traffic. An example of a worm that uses DoS attack is Code Red (Berghel 2001). It was programmed to unleash a DoS attack on the Whitehouse.gov website, targeting the actual Whitehouse.gov IP address.

IV. Destructive

This will cause harm to the computer or the host. According to Shannon and Moore (2004), the Witty worm deletes a randomly chosen section of the hard drive, which results in the computer becoming unusable. Viking worm is another example of a worm that infects executable files in both local drives and network shares, which harm to the victim’s computer (Anton 2009).

V. Phishing

Phishing is a criminal activity that employs social engineering techniques (Tsow 2006). Phishers attempt to acquire sensitive information fraudulently, such as usernames, passwords and credit card details, by presenting themselves as a trustworthy entity through electronic communication. Phishing can be undertaken through email or instant messaging and may ask the user to provide details of a website of which they are a member. Attempts to deal with the growing number of reported phishing incidents include legislation, user training and technical measures.

VI. Command and Control

Command and control refers to the capability of a worm to send important information, such as usernames and passwords, from the infected computer to the worm's writer via the Internet. This allows the worm's writer to remotely control any infected computer and Koobface is an example of such a worm.

VII. Infect Registry

The easiest way to ensure that a worm remains within a victim's computer is by hooking at a victim's computer registry. This is due to the fact that there are many registry entries that control the launching programme or service. Thus, to infect the windows operating system of a computer, the worm just has to drop itself at the registry. The most common entry where a worm would drop itself is *'Computer\ HKEY_Local_Machine\ Software\ Microsoft\ Windows\ CurrentVersion\ Run ':* this allows the worm to run when the computer boots up.

VIII. Mass Mailing

Mass mailing refers to a worm that is capable of sending itself to the email addresses found on an infected computer, using the victim's email client system or any other email client. Some mass mailing worms have their own SMTP email engine server, in order to ensure they succeed. Examples of mass mailing worms are Netsky and Mydoom.

IX. OS Version

Code Red II is an example of worm that has a different payload; thus, it relies on the operating system that it infects. For example, if an infected computer has a Chinese version of its operating system, the worm may produce up to 600 threads of propagation rate and then it may infects other systems for two days (Cisco 2004).

X. Metamorphic

The metamorphic worm has the same features as the polymorphic worm, where the code is programmed to change after a set duration of time. In addition, the functionality or the behaviour of the metamorphic worm is also programmed to change for a particular length of time, which is at odds with the polymorphic worm. This worm keeps on changing the code and its functionality for the purpose of avoiding being detected by anti-virus software.

XII. Apply Patch or Harden Configuration

The Nachi worm, also known as the W32.Welchia.Worm, spreads through and exploits the multiple vulnerabilities that exist within Windows operating system. Blaster worm is another example of a worm that downloads a Microsoft Windows update to a vulnerable computer and then removes the worm that already resided within the victim’s computer. These worms are then used and exploit the same vulnerabilities for the purpose of infecting the victim’s computer (Symantec 2003).

XIII. Degrade Performance

Once the worm succeeds in infecting the victim’s computer, it degrades normal computer performance and stability down to 80% from its normal condition.

D) Operating Algorithm

An operating algorithm is defined as a technique used by worms in order to avoid detection and Albanese *et al.* (2004) defined and classified the concept as a worm survival method. There are various categories of operating algorithm, as outlined below:

I. Polymorphic

A polymorphic worm changes all or part of their code each time an infected computer is rebooted and this helps the worm to avoid detection through the anti-virus scanning process. Kruegel *et al.* (2005) defined the polymorphic worm as a worm that is able to change its binary representation as part of the spreading process. This is done by employing self-encryption mechanisms or semantic-preserving code manipulation techniques. Consequently, a copy of a polymorphic worm may no longer share a common invariant substring of sufficient length and the existing systems will not recognise the worm's copy in the network streams.

II. Stealth

The stealth worm employs a concealment mechanism: it spreads slowly, evokes no irregular communication pattern and spreads in such a manner that detection proves difficult. Cheetancheri (1998) stated that the goal of the stealth worm is to spread to as many hosts as possible without being detected. However, once such a worm is detected, manual means of mitigation are possible.

III. Terminate and Stay Resident (TSR)

The terminate and stay resident (TSR) worm exploits a variety of techniques to remain resident in memory once the host programme that it infected is terminated. This kind of worm is also known as a resident or indirect worm, as it remains within the memory whilst searching for another file to infect.

IV. Anti Anti-virus

An anti anti-virus worm corrupts anti-virus software by deleting or changing anti-virus software and the data files, in order to ensure that the anti-virus software does not function properly. According to Nachenberg (2000), the anti anti-virus worm, also known as a retrovirus, is a computer virus that attacks anti-virus software in order to prevent itself from being detected. Retrovirus deletes anti-virus definition files, disables resident memory for anti-virus protection and attempts to disable anti-virus software in many ways.

E) Propagation

Propagation is a worm capability of spreading itself to another host or network and there are two ways in which such a worm can reproduce itself: through scanning or in a passive way.

I. Scanning

Scanning is a method employed by worms to find a victim, similar to the method proposed by Weaver *et al.* (2003). There are two possible scanning methods, which are random scanning and sequential scanning.

Random Scanning

This is the most popular scanning method, where the worm simply picks a random IP address from the network and then tries to connect to and infect it. An example of a random scanning worm is the Blaster worm (Bailey *et al.* 2005).

Sequential Scanning (Hitlist)

The worm releaser scans the network in advance and develops a complete hit list of all vulnerable systems on the network. The worm carries this address list with it and spreads throughout the list.

II. Passive

A worm that employs a passive monitoring technique does not actively search for new victims. Rather, it waits for a new target or relies on the user in discovering new targets. Christoffersen and Mauland (2006) asserted that the passive worm tends to have a slow propagation rate and is often difficult to detect because it generates modest anomalous reconnaissance traffic. Modest anomalous reconnaissance traffic means only small amount of abnormal scanning traffic is generated to the victim's computer, and most of the monitoring security tool will not assume it as a malicious activity since the quantity of the abnormal traffic is too small. For monitoring tool, the traffic has to reach certain limit in order for it to trigger any alert.

4.3.2 STAKCERT Worm Relational Model

Skoudis and Zeltser (2004) stated that one of the ways to prepare for a super worm is through the formation of a computer incident response team, with defined procedures for battling the worm. It is easier to confront a worm attack, if awareness of the threats posed by worms is taken into consideration. Unfortunately, it is hard to know what threats future worms will pose and thus it is important for us to know how to act upon the threats posed by any worm.

In order for organisations or users to defend their system or computer from the threat of a worm, the architecture and relationship with worm parameters and the environment should be well defined (Saudi *et al.* 2009, Saudi *et al.* 2010a). Ellis (2003) defined the worm relational model as the mathematical articulation of the relationship between the worm parameters, the current state of the environment and the subsequent state of the environment. Furthermore, Ellis (2003) presented a framework for the worm relational model that incorporated targeting, vulnerability, visibility and infectability. This is a well-structured relational model and is represented by relational algebra. An improvement that could be made to this relational model is by integrating the worm response so that it isolates itself if danger is apparent (also known as apoptosis), which is implemented in the STAKCERT worm model for worm response. By integrating the apoptosis for worm response, the worm will not propagate further. This model is related to worm parameters, attributes of the environment and the worm's subsequent potency. However, it is worth bearing in mind that the development of the STAKCERT worm relational model is based on the testing of the STAKCERT worm classification, using dynamic, static and statistical analyses. All the procedures and the details of static and dynamic analysis can be found in Chapter 3, Section 3.2.4.1.

Referring to the STAKCERT relational model, a frequency analysis was conducted to locate the highest frequently-occurring number to the lowest for each attribute that exists in this relational model. Then the relationship is verified, in terms of the STAKCERT relational model, by conducting the Chi-square and symmetric measure tests. Figure 4.4 below features the STAKCERT relational model.

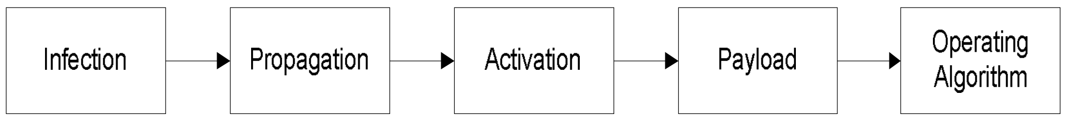


Figure 4.4. STAKCERT Relational Model.

With regards to frequency analysis, the top ten ways in which worms infect computer systems are identified, followed by the three main ways of propagation, the seven main methods of worm activation, the top ten payload types and the four main operating algorithm methods. All of these relationships can trigger the apoptosis condition and details of this condition can be found in Chapter 5. In the next section, the frequency analysis details are outlined and the Chi-square and symmetric measure tests are discussed.

4.4 Experimental Results on VX Heavens Datasets

The experimental results were divided into two categories, which are statistical analysis testing and the STAKCERT model for worm classification detection testing. The statistical analysis testing, which consisted of frequency analysis and the Chi-square and symmetric measures, was conducted in order to identify the highest frequency to the lowest frequency of the worm occurrence and to show that the features of and the relationship with the STAKCERT relational model did not occur by chance. Under the STAKCERT model for worm classification detection testing, clustering was initially conducted, in order to identify different types of worms from the datasets. Five different worms were identified as a result of clustering testing and these were later used as the input for classification detection testing. The classification detection testing was undertaken in order to demonstrate the accuracy of the STAKCERT model for worm detection and a comparison with other existing work was also undertaken.

4.4.1 Frequency Analysis

In order to identify the most important attributes of worm detection and to determine the relationship between these attributes, the frequency analysis and Chi-square and symmetric measure tests are conducted. In terms of frequency analysis, an analysis of the infection results showed that 27.3% of infection occurred through files, followed by email (9.9%). The rest of infection categories were sharing directories, file and sharing directories and file, email and vulnerability (representing 8.7%; 4.3% represented vulnerability and 3.1% each for file and vulnerability). Three categories (email, chatting channels and sharing directories) represented 2.5% vulnerability each and others were a combination of the different categories, in terms of infection. A few interesting associations were noted, in terms of the current way in which a worm infects and our findings. Based on the infection analysis results, as outlined in Figure 4.5, file, email, vulnerability and sharing directories are the most common methods of worm infection.

The top threats for January 2010, as presented by Eset (2010), were vulnerability, file and email. These are still being employed by worms in infecting victims' computers. As established by Eset (2010) paper, the Win32/Conficker worm exploits the vulnerabilities that exist within the Windows operating system, while the INF/Autorun worm uses the autorun.inf file to infect a system. The Win32/PSW.OnlineGames worm uses a phishing attack to steal information from games players who participate in online games and phishing can also rapidly spread through email. When the trend of how worms spread between the years 2001-2010 was analysed, it was ascertained that file, vulnerability and email were the most common methods of transport.

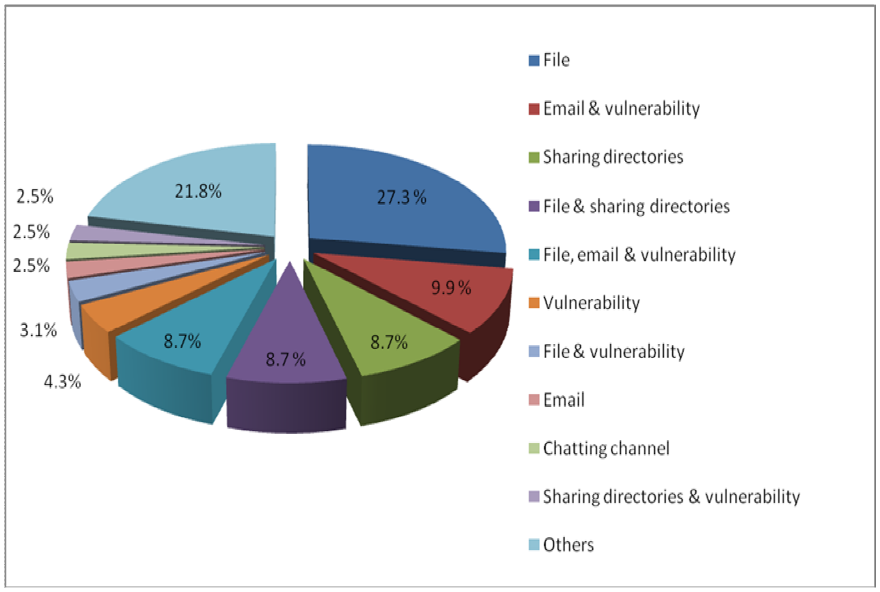


Figure 4.5. Analysis of Infection Results.

For the infection analysis, the relationship between file, vulnerability and email was explored in more depth and there was a scenario where the worm only infected via a file, email or vulnerability. Nevertheless, certain worms use two or three way combination of these to infect a victim's computer. Between 1971 and 2010, there were many methods of worm infection (Trend Micro 2008). Examples of worms exploiting vulnerabilities in websites or Windows operating system include the Code Red worm (2001), the Nimda worm (2001) and the Conficker worm (2008). However, the other sources of worm infection cannot simply be ignored. Chatting channels, social network websites, removable drives (such as USB), P2P (peer-to-peer) networks and smart phones are alternative sources of worm infection and are becoming increasingly so. Worm_Autorun.AZ is an example of a worm that spreads via chatting channels, P2P networks and removable drives.

In terms of the analysis of the propagation results, only 10% incorporated random scanning, followed by 3% sequence scanning: the remainder had no scanning implications. This analysis is outlined in Figure 4.6. Once a worm has infected a victim's computer, it needs to spread itself to another computer or network. However, based on the testing results with the datasets, more than 50% of worms did not propagate themselves.

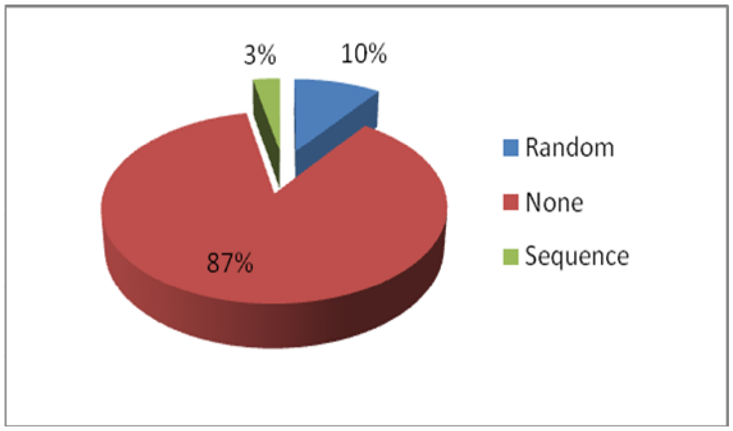


Figure 4.6. Analysis of Propagation Results.

The question that is thus raised is: should propagation be highlighted as one of the important components in classifying worms? Even though random and sequence propagation represents only 10% and 3% of worms respectively, we cannot underestimate these methods of propagation. Worms such as Code Red, Nimda, Blaster, Nachi and Sobig.F have their own propagation rate (Saudi 2005). Moreover, based on this thesis analysis, propagation is one of the most important elements in detecting a worm attack.

In terms of the analysis of the activation results (as shown in Figure 4.7), more than half of worms (54.8%) were self activated, while others were activated through a combination of self activation and a human trigger (at 21.7% and 18% respectively).

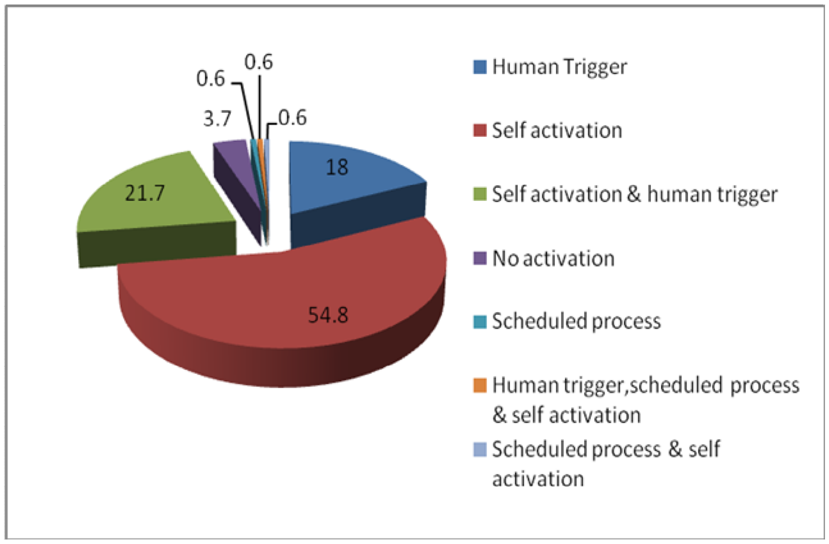


Figure 4.7. Analysis of Activation Results.

No activation accounted for 3.7% of worms, with the remainder of the factors representing 0.6% each. Self activation refers to the ability of worms to spread themselves to other computers without the need for human intervention; i.e., the Conficker worm, which exploits vulnerabilities in Microsoft programmes. The human trigger is implemented by several factors, such as social engineering techniques, logging onto certain websites or downloading certain files, which leads to file or script execution or the opening of certain ports on the victim's computer. There are different ways how worm activation is triggered have been identified in this thesis and based on the analysis conducted, activation is considered as one of the important characteristics in worm detection.

Figure 4.8 displays the top ten types of payload: Destructive implication yielded a figure of 14.3%, while performance degradation came second, at 9.3%. The autorun registry was third, at 5%, and the combination of backdoor and autorun registry yielded a figure of 1.9%.

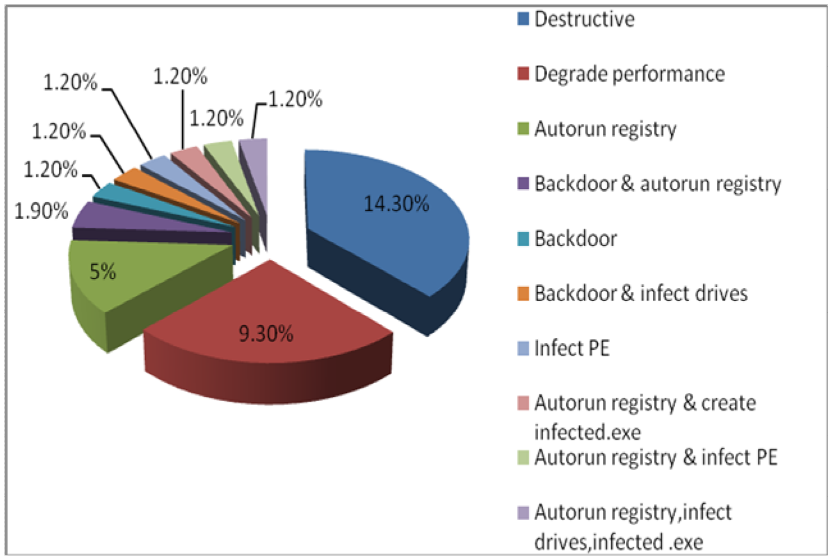


Figure 4.8. Analysis of Top 10 Payload Results.

The rest, which are backdoor, infect PE executable, the combination of backdoor and drives infection, the combination of the autorun registry, the creation of infected .exe, the combination of autorun registry, drive infection and the creation of infected .exe, represented 1.2% each. Other payloads not discussed here are mostly based on a combination of the different payloads. The target towards the end of this research is to produce a STAKCERT model for worm detection and response and payload is seen as one of the important elements being incorporated as input for this model. There have been so many payloads identified as a result of conducted research and in the STAKCERT model, the STAKCERT worm classification is used as the basis and thus it is important to ensure that each component is well tested. It is interesting to note that all the features selected are related to each other, based on the static, dynamic and statistical analyses. This shows that the STAKCERT worm classification proposed in this thesis is useful and plays significant role for worm detection.

Last but not least is the operating algorithm, which refers to the technique employed by worms in order to avoid detection. The operating algorithm is considered an added feature that should be taken into account when building up a STAKCERT model because it is important to know the features integrated by a worm to avoid from being detected. As a result of the conducted tests, it was ascertained that a majority of 96% of worms were categorised as terminate and stay resident (TSR) as displayed in Figure 4.9.

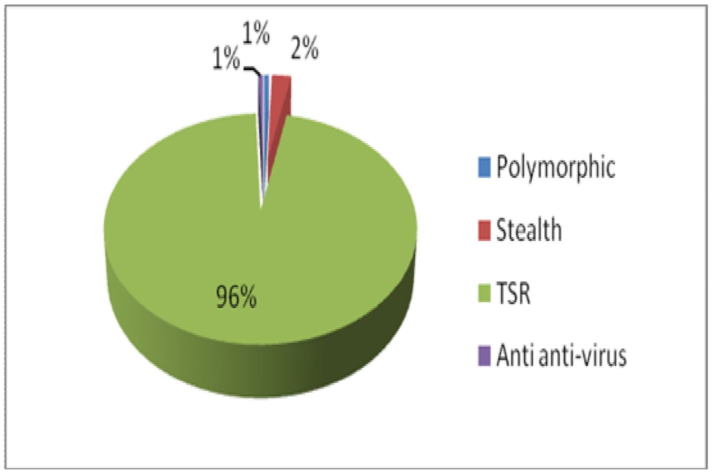


Figure 4.9. Analysis of Operating Algorithm Results.

Stealth referred to 2% of worms, followed by the polymorphic and anti anti-virus worms, at 1% each. Each of the operating algorithm has its own method of spreading and replicating to other computers. Many researchers within the worm field focus on the polymorphic worm, but still the other techniques should not be ignored. If, in the near future, a worm uses a combination of polymorphic, stealth, TSR and anti anti-virus to conceal itself, an in-depth study regarding this new features should be carried out, so a good solution to detect this worm can be developed. If a good understanding of how each of these techniques works is established, it is possible to produce a defensive mechanism using such methods in combination.

Based on the analysis of the tests conducted, it can be concluded that each of the features in question are related to one another. The formation of the STAKCERT relational model is based on the premise that each feature plays an important role in worm detection and isolation and supports the relevance of current issues related to worm infection.

4.4.2 Chi-square and Symmetric Measure Results

The formation of the STAKCERT relational model is based on the features of the STAKCERT worm classification. Previously, under frequency analysis, the importance of each feature was identified and this generally gave an idea of the relationship between the features. To support this, the Chi-square and symmetric measure tests are conducted, in order to determine the relationship between the features. A detailed explanation of the Chi-square and symmetric measure definitions, equations and purposes can be found in Chapter 3, section 3.2.4.2. Only three of the main features (infection, activation and payload) were tested using the Chi-square and symmetric measure tests, as the other features did not meet the testing requirements of the Chi-square tests. However, this should not be a problem as the frequency analysis and the Fisher's exact test have been conducted. The Chi-square test becomes invalid if the expected frequency is less than 5. If the expected counts for the nominal data are less than five, with the condition that it is a 2x2 contingency table (the number of degrees of freedom is always 1), the alternative test that can be carried out is known as Fisher's exact test. This Fisher's exact test has the same objective as the Chi-square test, but it is dedicated to expected counts of less than five. Using the p value of 0.05 for both tests yielded the result that most of the features showed a statistically significant relationship and details of the tests and other further information can be found in Appendix A. Based on Chi-square, symmetric measures and Fisher's test findings, it can be concluded that each relationship has its own representation and interpretation. For subsequent analyses, 160 datasets resulting from these findings are further analysed and tested.

4.4.3 STAKCERT Model for Worm Detection Results

Figure 4.10 shows an overview of how the datasets were clustered and classified, once the feature selection process was completed. Features selection is part of the STAKCERT KDD processes and thus all the datasets were previously tested, using the STAKCERT relational model as the basis for this. This is later used as the input for the clustering and classification processes.

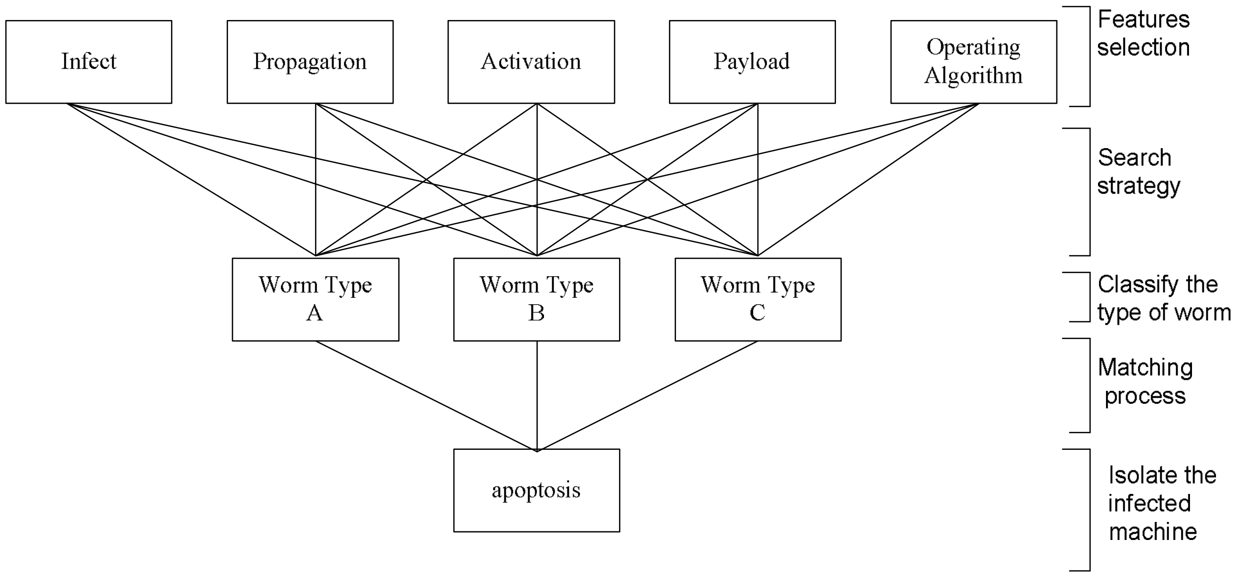


Figure 4.10. An Overview of Worm Clustering and Classification.

4.4.3.1 STAKCERT Worm Clustering

In a test that was conducted using WEKA software, the datasets retrieved from the 160 datasets where each dataset has five main features: infection, propagation, activation, payload and operating algorithm, were clustered using simple k-means. The clustering was first conducted to discover a new set of worm categories from the datasets. The datasets retrieved from the VX Heavens consisted of thousands of data items that were not yet clustered or classified. Thus, clustering was conducted in order to group all the datasets into different groups of worms. Once the clustering was completed, then the classification between predicted and actual different groups of worm can be carried out. If this clustering was not carried out, it is hard to conduct the classification testing. The datasets were clustered using the k-means algorithm, with five sets of clusters, ten random seeds and using Euclidean distance as a metric. The k-means is chosen due to its effectiveness, where the datasets are partitioned based on centroids (also known as mean) and then the datasets are assigned to their closest cluster centre based on Euclidean distance. The whole process is repeated with a new cluster centre until the same point is assigned to each cluster. Details of how k-means works can be found in Chapter 3, Section 3.2.4.4.

In terms of this clustering, cluster 1 accounted for 46% of the datasets, followed by cluster 2 at 19%, cluster 3 at 15%, cluster 4 at 11% and cluster 5 at 9% (see Figure 4.11). Cluster 1 is also known as worm type I, whilst cluster 2 is also known as worm type 2, cluster 3 as worm type 3, cluster 4 as worm type 4 and cluster 5 as worm type 5. Prior to the clustering method; static, dynamic and statistical analyses were conducted, in order to verify the relationship between the five main features used as variables in the clustering method. All the results related with the static, dynamic and statistical analyses are already explained under subsection 4.41 and 4.42 and can be found in the paper published by Saudi *et al.* (2010a, 2010b). The details of the clustering results and the details of the different types of worm categorised as worm types 1-5 can be found in Appendix B.

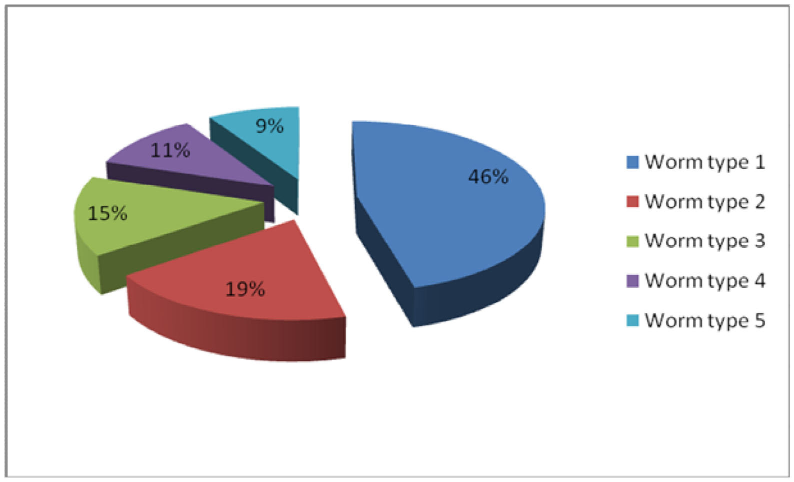


Figure 4.11. Worms Clustering.

Once the clustering processes were completed, classification was undertaken. The clustered worms were classified using five different algorithms (which were the Multilayer Perceptron (MLP), Sequential Minimal Optimisation (SMO), Naïve Bayes, J48 and IBk) and were tested using the 10-fold cross validation test. In order to identify the most accurate classification algorithm, WEKA is used by running five different algorithms.

4.3.2 Results Summary

In terms of the tests conducted, the configuration used for the different algorithms can be found in Chapter 3, Section 3.2.4.4 (Table 3.7). Figure 4.12 shows the percentages correctly classified or known as the overall accuracy by these five different algorithms. The Multilayer Perceptron has the highest accuracy, followed by SMO, IBk, Naïve Bayes and J48.

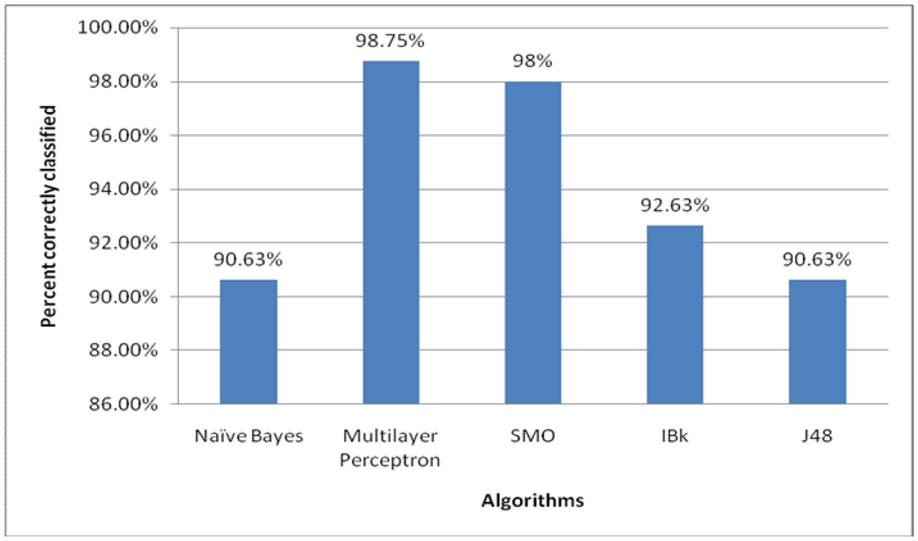


Figure 4.12. Percentage Correctly Classified by Different Algorithms.

As the datasets were nominal, the performance criteria of the STAKCERT model for worm detection focused on the accuracy of the correctly classified and incorrectly classified. In addition to this, other performance criteria (TP Rate, FP Rate, FN Rate, Precision, Recall and F-measure) were also discussed, in order to get a clearer picture of the output results. Details of the definitions and the equations of the above performance criteria terms can be found in Chapter 3, Section 3.2.4.4 (entitled 'Performance Criteria Definitions').

The results of the five different algorithms used for testing are summarised in Table 4.1. Thus, Table 4.1 outlines detection accuracy, based on the TP Rate, the FP Rate, the FN Rate, overall accuracy for the Multilayer Perceptron (MLP), SMO, IBk, Naïve Bayes and J48. As displayed in Table 4.1, MLP outperformed the other algorithms; thus the results for the MLP are explained in detail. The interpretations of the other algorithms were similar to the MLP algorithm, but had a different analysis conclusion. Furthermore, how each of the performance criteria was calculated and the meaning of each value was presented, are explained in the next subsection.

Referring to Table 4.1, there are four main characteristics presented, which are TP Rate (TPR), FP Rate (FPR), FN Rate (FNR) and overall accuracy (OA). These four main performance criteria were chosen as they represented the most important features in verifying the classifier algorithm for worm detection.

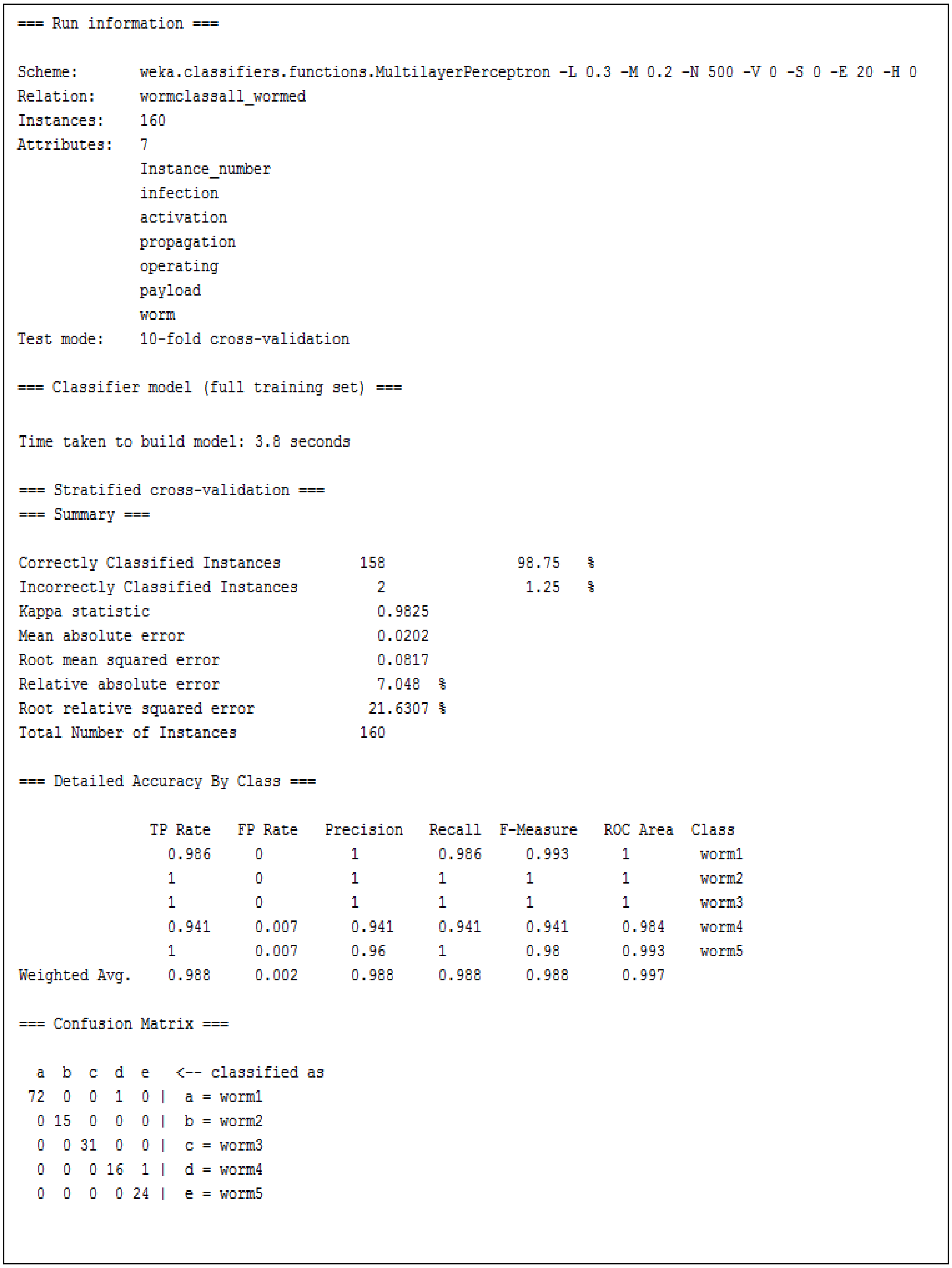
Table 4.1. Summarisation of the Results for All Algorithms.

	Multilayer Perceptron				SMO				IBk				Naïve Bayes				J48			
	TPR	FPR	FNR	OA	TPR	FPR	FNR	OA	TPR	FPR	FNR	OA	TPR	FPR	FNR	OA	TPR	FPR	FNR	OA
Average result in %	98.8	0.2	1.45	98.75	98.1	0.2	2.63	98.13	93.1	2.8	8.93	93.13	90.6	3.3	9.84	90.63	90.6	6.2	17.6	90.63
Worm1 (%)	98.6	0	1.37	99.38	98.6	0	1.37	99.38	94.5	4.6	5.48	95	87.7	4.6	12.33	91.88	98.6	12.6	1.37	92.15
Worm2 (%)	100	0	0	100	100	0.7	0	99.38	80	0	20	98.13	86.7	0	13.33	98.75	86.7	0.7	13.3	98.13
Worm3 (%)	100	0	0	100	100	0	0	100	96.8	0.8	3.23	98.75	100	2.3	0	98.12	100	0.8	0	99.38
Worm4 (%)	94.1	0.7	5.88	98.75	88.2	0.7	11.77	98.13	88.2	2.1	11.77	96.88	76.5	3.5	23.53	94.38	29.4	0	70.59	92.15
Worm5 (%)	100	0.7	0	99.38	100	0.7	0	99.38	95.8	2.2	4.17	97.5	100	2.2	0	98.13	100	1.5	0	98.75

** TPR = True Positive Rate (also known as detection accuracy), FPR = False Positive Rate, OA = Overall Accuracy, FNR= False Negative Rate*

4.4.3.2.1 Multilayer Perceptron Findings

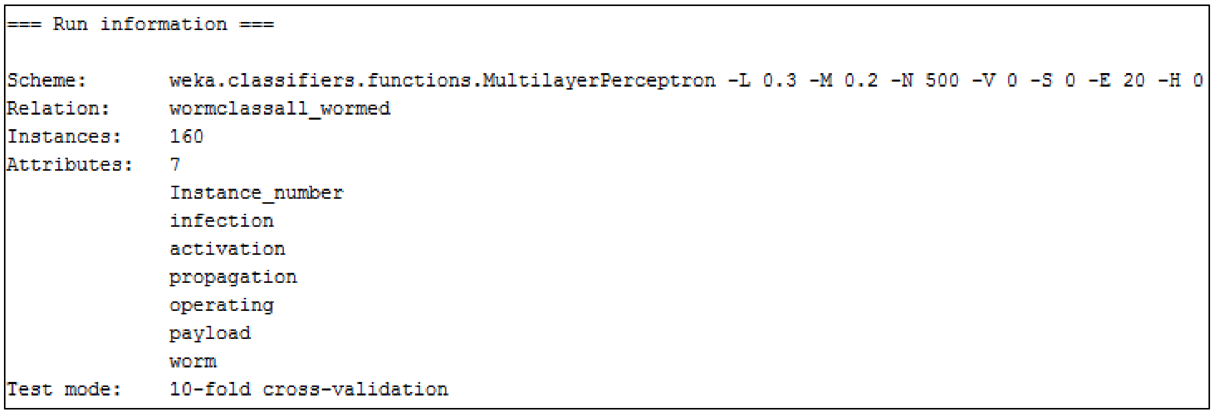
In this section, detailed results of the Multilayer Perceptron (MLP) algorithm are presented. All of the outputs were generated using the WEKA. It is open source and JAVA based. Once the outputs are already being analysed and understood, it could be concluded whether the predicted results were the same as the actual results. Based on this thesis test results and comparing the predicted results with the actual results, the Multilayer Perceptron algorithm demonstrated the highest performance of all the algorithms and Figure 4.13 displays the results for this.



```
=== Run information ===

Scheme:       weka.classifiers.functions.MultilayerPerceptron -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H 0
Relation:     wormclassall_wormed
Instances:    160
Attributes:   7
              Instance_number
              infection
              activation
              propagation
              operating
              payload
              worm
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===


Time taken to build model: 3.8 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         158               98.75   %
Incorrectly Classified Instances         2                1.25   %
Kappa statistic                          0.9825
Mean absolute error                      0.0202
Root mean squared error                  0.0817
Relative absolute error                  7.048  %
Root relative squared error             21.6307 %
Total Number of Instances              160

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.986     0          1         0.986     0.993      1        worm1
                 1         0          1         1         1          1        worm2
                 1         0          1         1         1          1        worm3
                 0.941     0.007      0.941     0.941     0.941      0.984    worm4
                 1         0.007      0.96      1         0.98       0.993    worm5
Weighted Avg.    0.988     0.002      0.988     0.988     0.988      0.997

=== Confusion Matrix ===

  a  b  c  d  e   <-- classified as
 72  0  0  1  0 |  a = worm1
  0 15  0  0  0 |  b = worm2
  0  0 31  0  0 |  c = worm3
  0  0  0 16  1 |  d = worm4
  0  0  0  0 24 |  e = worm5
```

Figure 4.13. Multilayer Perceptron Results.

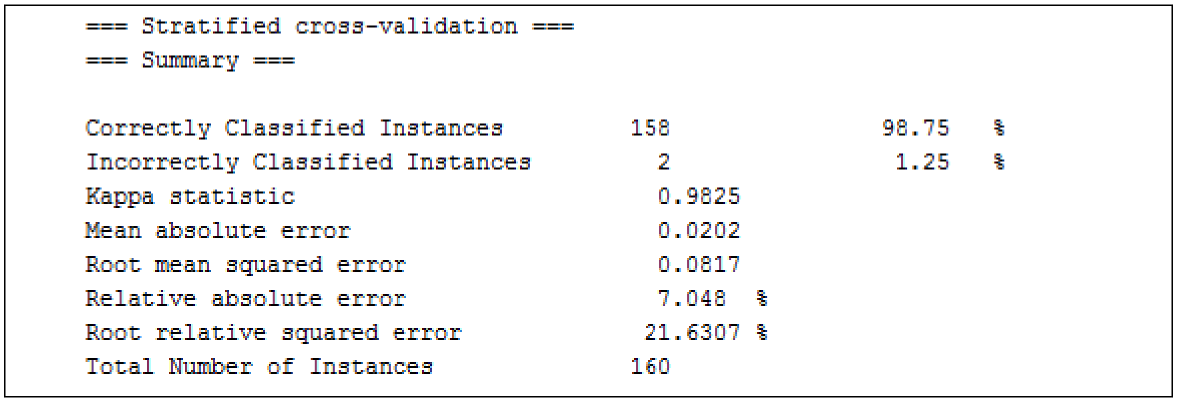
Next, the outputs from Figure 4.13 are explained in detail.



```
=== Run information ===

Scheme:       weka.classifiers.functions.MultilayerPerceptron -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H 0
Relation:     wormclassall_wormed
Instances:    160
Attributes:   7
              Instance_number
              infection
              activation
              propagation
              operating
              payload
              worm
Test mode:    10-fold cross-validation
```

Figure 4.14. Extracted Output 1 from MLP Results.

The above extracted output (Figure 4.14) is the configuration setting for the Multilayer Perceptron algorithm. The first line shows that, for this testing, the learning scheme was ‘*weka.classifiers.functions.MultilayerPerceptron*’ or the neural network algorithm: this uses backpropagation to classify the datasets. The first line shows ‘scheme’, where the parameters are shown as ‘—L 0.3 –M 0.2 –N 500 –V 0 –S 0 –E 20 –H 0’, which states that the learning rate is equal to 0.3, momentum is equal to 0.2, training time is 500, zero validation set size, zero seed, validation threshold is equal to 20 and the hidden layer is zero. The second line shows the file used for testing and the third line shows there are 160 instances involved in this testing. On the next line, there are seven main attributes, which are the instance number (for numbering), infection, activation, propagation, operating, payload and worm. The ‘test mode’ used was the 10-fold cross validation.

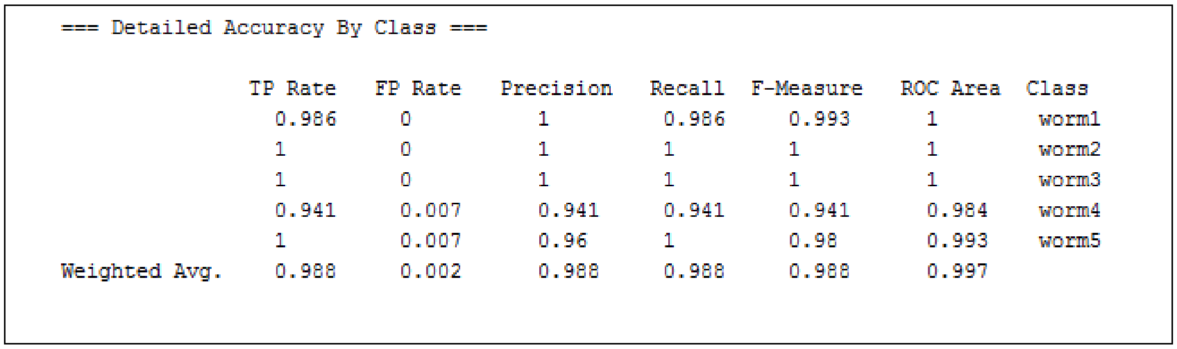


```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         158               98.75   %
Incorrectly Classified Instances         2                1.25   %
Kappa statistic                          0.9825
Mean absolute error                      0.0202
Root mean squared error                  0.0817
Relative absolute error                  7.048  %
Root relative squared error             21.6307 %
Total Number of Instances              160
```

Figure 4.15. Extracted Output 2 from MLP Results.

The extracted output in Figure 4.15 was among the most important aspects of verifying the classifier performance. The first 2 lines are the most useful, as our class variable is nominal. The first line shows the number and percentage of cases that were correctly classified (also known as accuracy) and the accuracy for this classifier was 158 (98.75%). For the incorrectly classified, there were 2 cases at 1.25% and the Kappa statistic shows that the 0.9825 and 98.25% predictions within the actual classes are correlated. The Kappa statistic was used to measure the agreement of predictions with the actual class; the nearer the Kappa statistic is to the value of 1, the stronger the correlation between predictions and actual classes. The next few lines show the error values for this testing but were not taken into account as our testing only involved the nominal classes and classification tasks. Furthermore, these values are applicable, yet error values would be reasonable criteria if it were involved with regression testing.

This following extract (Figure 4.16) is the detailed accuracy results for all worm classes that were extracted from Figure 4.13.

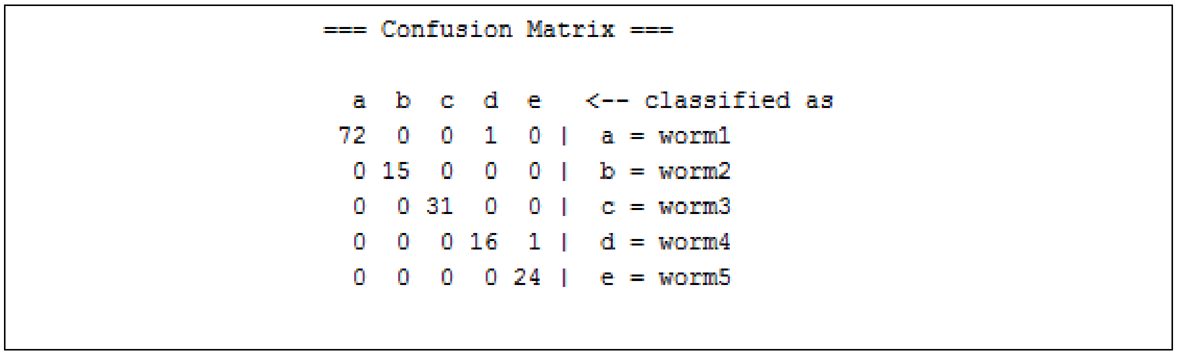


```
=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.986     0          1         0.986     0.993      1         worm1
                 1         0          1         1         1          1         worm2
                 1         0          1         1         1          1         worm3
                 0.941     0.007      0.941     0.941     0.941      0.984     worm4
                 1         0.007      0.96      1         0.98       0.993     worm5
Weighted Avg.    0.988     0.002      0.988     0.988     0.988      0.997
```

Figure 4.16. Extracted Output 3 from MLP Results.

The first two columns in Figure 4.16 are the TP Rate (true positive rate) and the FP Rate (False Positive Rate), followed by Precision, Recall, F-Measure, ROC Area and Class. The TP Rate is the ratio of predicted correctly classified cases (as worm 1, worm 2, worm 3, worm 4 and worm 5) to the total of positive cases. The FP Rate is the ratio of predicted incorrectly classified cases (as worm 1, worm 2, worm 3, worm 4 and worm 5) to the total of incorrectly classified cases and correctly classified as the wrong cases. Precision refers to the proportion of cases that are correctly classified as worm 1, worm 2, worm 3, worm 4 and worm 5 from the all the cases being classified for the dedicated classes of worm 1, worm 2, worm 3, worm 4 and worm 5. The recall is equivalent to the TP Rate and F-Measure is a combined measure of Precision and Recall. The ROC area is based upon the TP rate and the FP rate and the Weighted Avg. refers to the average values for the five different worm classes.

Referring to Figure 4.16, the ROC area represents the area under the ROC curve and it can be concluded that, the nearer the ROC area value to 1, the more accurate the prediction of the classifier correctly classified. This was based on the average of the ROC area value of 0.997, with a FP Rate of 0.002 and a TP Rate of 0.988.



```
=== Confusion Matrix ===

  a  b  c  d  e   <-- classified as
 72  0  0  1  0 |  a = worm1
  0 15  0  0  0 |  b = worm2
  0  0 31  0  0 |  c = worm3
  0  0  0 16  1 |  d = worm4
  0  0  0  0 24 |  e = worm5
```

Figure 4.17. Extracted Output 4 from MLP Results.

A confusion matrix is a simple way of displaying the results of the experiments and is also known as a contingency table. In this testing, there were 5 classes (worm 1, worm 2, worm 3, worm 4 and worm 5) and thus a 5 x 5 confusion matrix was formed (as displayed in Figure 4.17). The rows of this confusion matrix represent the actual classes, while the columns represent the prediction classes. The predicted numbers of correctly classified instances are the sum of diagonals in the matrix i.e. 72+15+31+16+24=158. The other numbers from these diagonals represent the incorrectly classified; for example, for worm 1, (based on the confusion matrix 5x5 in Figure 4.17) the output values were calculated in the following way:

TP represents True Positive, TN represents True Negative, FP represents False Positive and FN represents False Negative. The values for TP = 72, TN = 0, FP = 0 and FN = 1.

TP Rate *= TP / (TP+FN)*
= 72 / (72 + 1)
= 0.986

FP Rate *= FP / (FP+TN)*
= 0 / (0 +87)
= 0

Precision*= TP / (TP+FP)*
= 72 / (72+0)
= 1

Accuracy*= (TP+TN)/ (TP+TN+FP+FN)*
= (72 + 87) / (72 +87+0+ 1)
= 0.9938

Error rate *= 1- Accuracy*
= 1- 0.9938
= 0.0062

F-measure *= 2 * recall * precision / (recall + precision)*
*= 2 * 0.986*1 /(0.986 +1)*
= 0.993

Recall **=** *TP Rate*
= 0.986

In referring to the confusion matrix in Figure 4.17, (for worm 1), there were 72 correctly classified (TP=72), 87 were correctly classified not as worm 1 (TN=87), none from the other cases of different classes were wrongly classified as worm 1(FP=0) and 1 from class worm 1 was wrongly classified (FN=1). Thus, the TP rate was 0.986, the FP Rate was 0 and the FN Rate was 0.0137. Precision was 1. Recall equivalents to TP rate was 0.986 and the F-Measure was 0.993. The ROC area was 1. Based on the TP rate, which was almost 1, the FP rate and the FN Rate was 0 and the ROC area was 1: this showed that worm 1 was correctly classified, with an accuracy of 99.38%, and the classifier prediction was likely to be the actual class of worm 1.

The rest of the calculations for the different classes of worms 2 to 5 used the same equations as above. For worm 2, there were 15 correctly classified (TP=15), while 145 were correctly classified as not worm 2 (TN=145). None from the other cases of different classes were wrongly classified as worm 2 (FP=0) and zero from worm 2 were wrongly classified (FN=0). Thus, TP rate was 0.986, while the FP rate and FN rate were 0 and Precision was 1. The Recall equivalent to TP rate was 1, as were the F-measure and the ROC area. It can be thus concluded that worm 2 was perfectly classified, based on the TP rate value (which was 100%). The FP rate and FN rate were 0% and the ROC area was 1. The classifier prediction was 100% correct, compared to the actual class of worm 2.

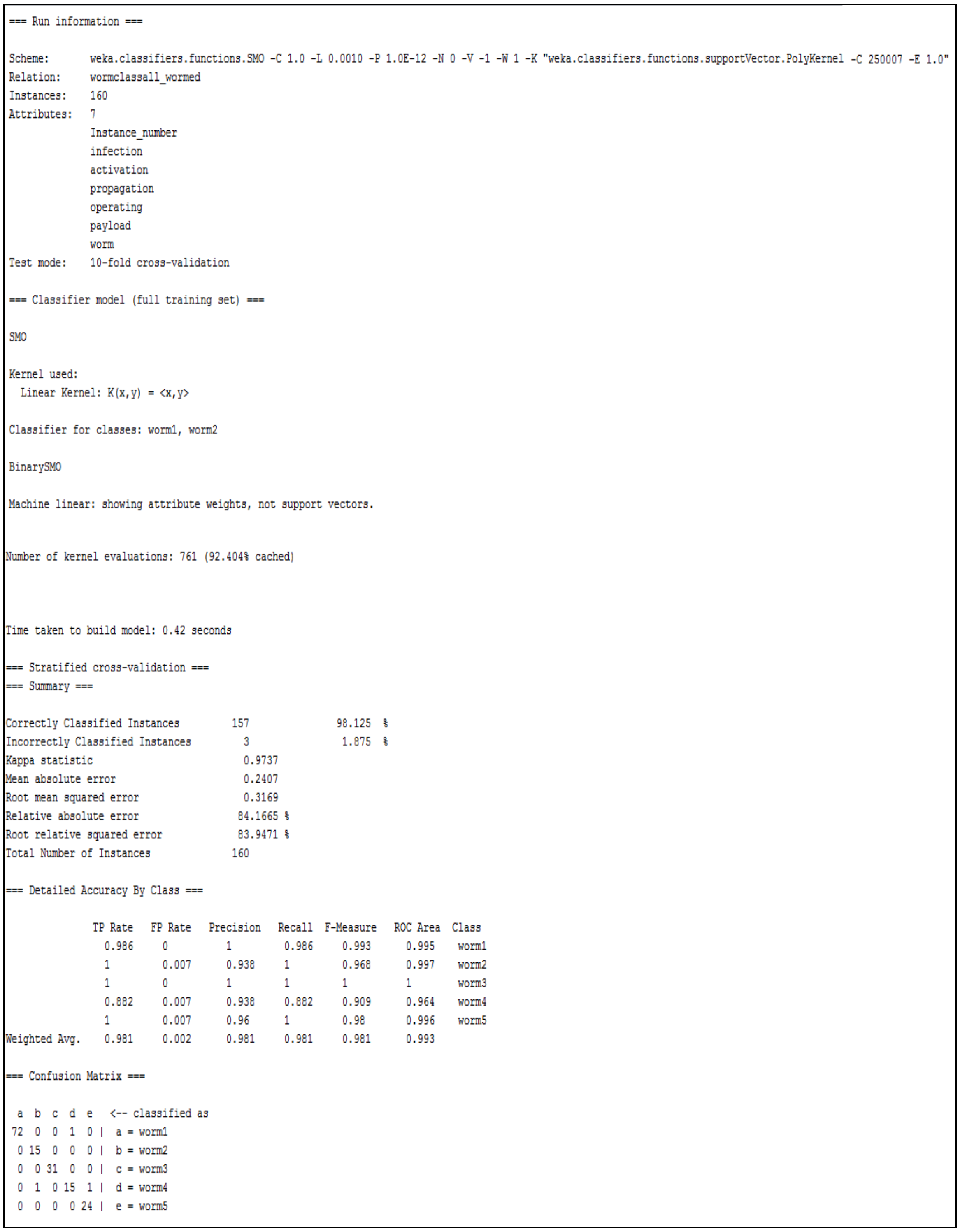
For worm 3, there were 31 correctly classified (TP=31), 0 from class worm 3 were wrongly classified (FN=0), 129 were correctly classified as not worm 3 (TN=129) and none from the other cases of different classes were wrongly classified as worm 3 (FP=0). TP rate was 31/(31+0)=1, FP rate was 0 and the FN rate was 0. Precision was 31/(31+0) =1, while recall equivalents to TP rate was 1. The F-Measure was (2 *1 *1)/(1+1) =1. It can be concluded that worm 3 was perfectly classified, as the TP rate value was 100%, the FP rate and FN rate were 0% and the ROC area was 1. The classifier prediction was 100% correct, compared to the actual class of worm 3.

In terms of worm 4, there were 16 correctly classified (TP=16), 1 from class worm 4 was wrongly classified (FN=1), 142 were correctly classified as not worm 2 (TN=142) and 1 from other cases of different classes was wrongly classified as worm 4(FP=1). TP rate was 16/(16+1)=0.941, FP rate was 1/(1+142)=0.007 and the FN Rate was 0.588. Precision was 16/(16+1)=0.941, while Recall equivalents to TPR was 0.941. F-Measure was (2 *0.941 *0.941)/(0.941+0.941)=0.941. Based on the TP rate (94.1%), the FP rate (0.7%), the FN Rate (5.88%) and the ROC area (0.984), it was shown that worm 4 was correctly classified: accuracy was 98.75%, compared to the actual class of worm 4.

For worm 5, there were 24 correctly classified (TP=24), 0 from class worm 5 were wrongly classified (FN=0), 135 were correctly classified as not worm 5 (TN=135) and 1 from the other cases of different classes was wrongly classified as worm 5 (FP=1). TP rate was 24/(24+0)=1, FP Rate=1/(1+135)=0.007 and the FN rate was 0. Precision was 24/(24+1)=0.96, while recall equivalents to TP rate was 1. The F-Measure was (2 *1 *0.96)/(1+0.96)=0.98. Based on the TP rate (96%), the FP rate (0.7%), the FN Rate (0%) and the ROC area (0.993), it was ascertained that worm 5 was correctly classified: accuracy was 99.38%, compared to the actual class of worm 5.

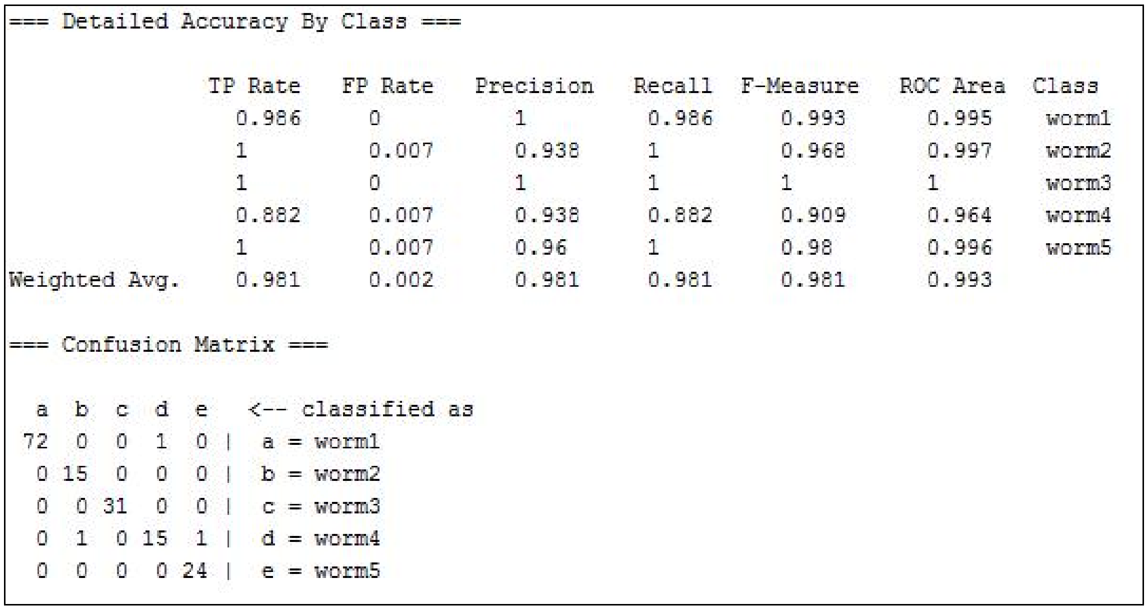
4.4.3.2.2 SMO Findings

In this section, a detailed explanation of the Sequential Minimal Optimisation (SMO) algorithm is presented. The SMO algorithm yielded the second highest overall performance, with an overall accuracy of 98.13% and a FP rate of 0.2%. The average of the TP rate for the five different classes was 98.1% and 0.2% represented a FP rate. Referring to the TP rate for each class under the 'detailed accuracy by class' heading in Figure 4.18 , it can be seen that worm 1 was 98.6%, worm 2, worm 3 and worm 5 were all 100% and worm 4 was 88.2%. The FP rate for worm 1 and worm 3 was 0% and worm 2, worm 4 and worm 5 had a FP rate of 0.7%. Note that, this thesis only discussed the TP rate, the FP rate, the FN rate, the ROC area and accuracy, as these five main performance criteria play an important role in verifying classifier algorithm performance. If the ROC area has the same value, in identifying the highest worm class performance, then the accuracy of each class is referred.



```
=== Run information ===

Scheme:       weka.classifiers.functions.SMO -C 1.0 -L 0.0010 -P 1.0E-12 -N 0 -V -1 -W 1 -K "weka.classifiers.functions.supportVector.PolyKernel -C 250007 -E 1.0"
Relation:     wormclassall_wormed
Instances:    160
Attributes:   7
              Instance_number
              infection
              activation
              propagation
              operating
              payload
              worm
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

SMO

Kernel used:
  Linear Kernel: K(x,y) = <x,y>

Classifier for classes: worm1, worm2

BinarySMO

Machine linear: showing attribute weights, not support vectors.


Number of kernel evaluations: 761 (92.404% cached)



Time taken to build model: 0.42 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         157               98.125  %
Incorrectly Classified Instances         3                1.875  %
Kappa statistic                          0.9737
Mean absolute error                      0.2407
Root mean squared error                  0.3169
Relative absolute error                 84.1665 %
Root relative squared error             83.9471 %
Total Number of Instances              160

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.986     0          1         0.986     0.993      0.995    worm1
                 1         0.007      0.938     1         0.968      0.997    worm2
                 1         0          1         1         1          1        worm3
                 0.882     0.007      0.938     0.882     0.909      0.964    worm4
                 1         0.007      0.96      1         0.98       0.996    worm5
Weighted Avg.    0.981     0.002      0.981     0.981     0.981      0.993

=== Confusion Matrix ===

  a  b  c  d  e   <-- classified as
 72  0  0  1  0 |  a = worm1
  0 15  0  0  0 |  b = worm2
  0  0 31  0  0 |  c = worm3
  0  1  0 15  1 |  d = worm4
  0  0  0  0 24 |  e = worm5
```

Figure 4.18. SMO Results.



```
=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.986     0         1          0.986     0.993      0.995    worm1
                 1         0.007     0.938      1         0.968      0.997    worm2
                 1         0         1          1         1          1        worm3
                 0.882     0.007     0.938      0.882     0.909      0.964    worm4
                 1         0.007     0.96       1         0.98       0.996    worm5
Weighted Avg.    0.981     0.002     0.981      0.981     0.981      0.993

=== Confusion Matrix ===

  a  b  c  d  e   <-- classified as
 72  0  0  1  0 |  a = worm1
  0 15  0  0  0 |  b = worm2
  0  0 31  0  0 |  c = worm3
  0  1  0 15  1 |  d = worm4
  0  0  0  0 24 |  e = worm5
```

Figure 4.19. Extracted Output from SMO Results.

The extracted outputs from the SMO results which consist of ‘detailed accuracy by class’ and ‘confusion matrix’ can be found in Figure 4.19.

In terms of the analysis of the ROC curve, a high result for the TP rate and a low result for the FP rate are good indicators of the produced predicted classifier result. The ROC area of worm 1 was 0.995, worm 2 was 0.997, worm 3 was 1, worm 4 was 0.964 and worm 5 was 0.996 and the accuracy for each worm class was 99.38% for worm 1, 99.38% for worm 2, 100% for worm 3, 98.13% for worm 4 and 99.38% for worm 5. The FN rate for each worm class was 1.37% for worm 1, 0% for worm 2, worm 3 and worm 5 and 11.77% for worm 4.

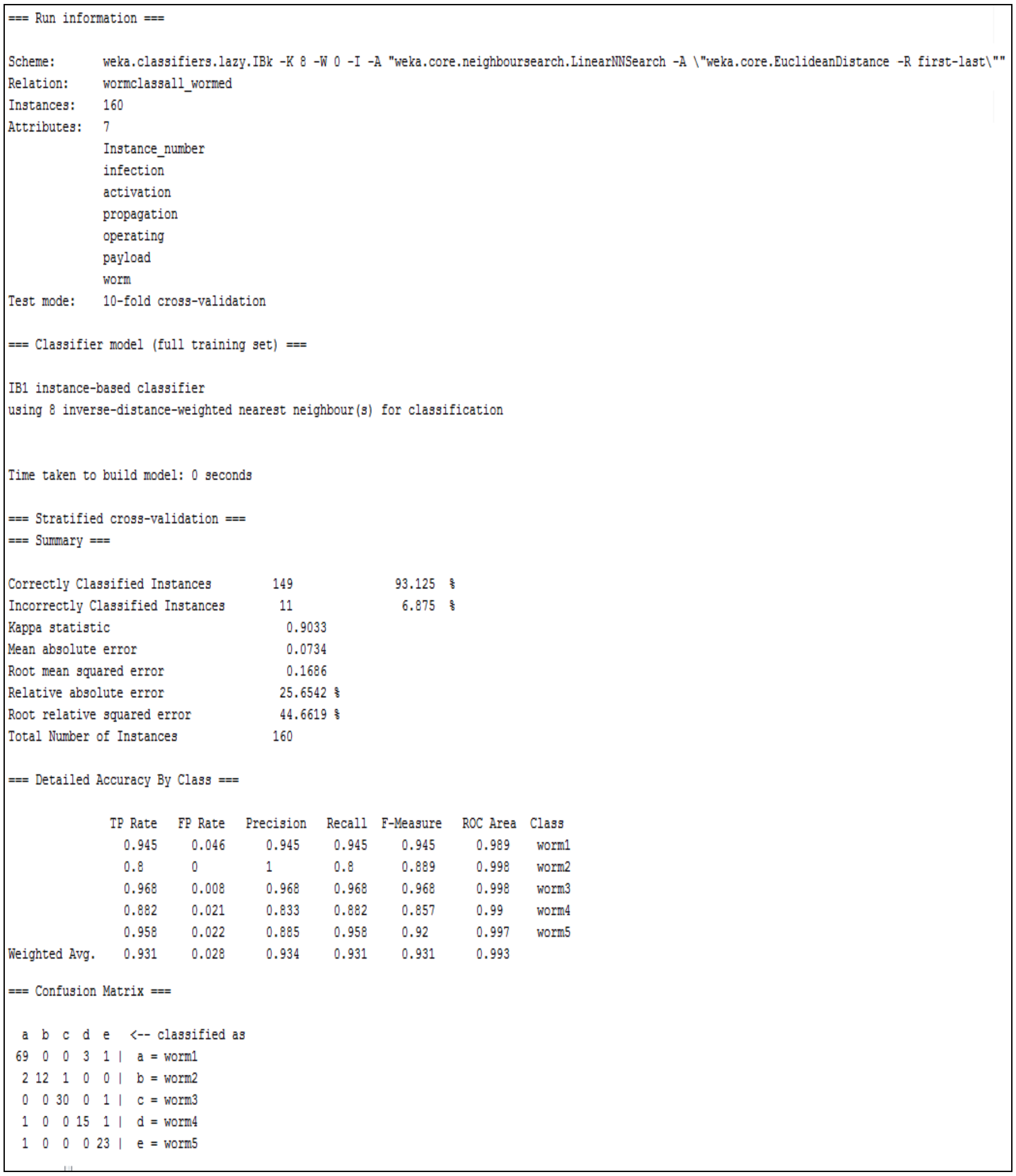
As mentioned earlier, the nearer the ROC area value to 1 indicates a better performance. Worm 3 had the highest performance, with a TP rate of 100%, a FP rate and FN rate of 0% and an accuracy rate of 100%. The prediction classifier was 100% just like the actual classifier.

4.4.3.2.3 IBk Findings

In this section, the detailed results for the IBk algorithm are presented (the IBk is the k-nearest neighbour classifier). The IBk algorithm was third ranking for overall performance, with an overall accuracy of 93.13%, an average FP rate of 2.8% and an average TP rate of 93.1%. Referring to the ‘detailed accuracy by class’ results in Figure 4.20, the TP rate for worm 1 was 94.5%, 80% for worm 2, 96.8% for worm 3, 88.2% for worm 4 and 95.8% for worm 5. The FP rate for worm 1 was 4.6%, 0% for worm 2, 0.8% for worm 3, 2.1% for worm 4 and 2.2% for worm 5. The FN rate for each worm class was 5.48% for worm 1, 20% for worm 2, 3.23% for worm 3, 11.77% for worm 4 and 4.17% for worm 5.

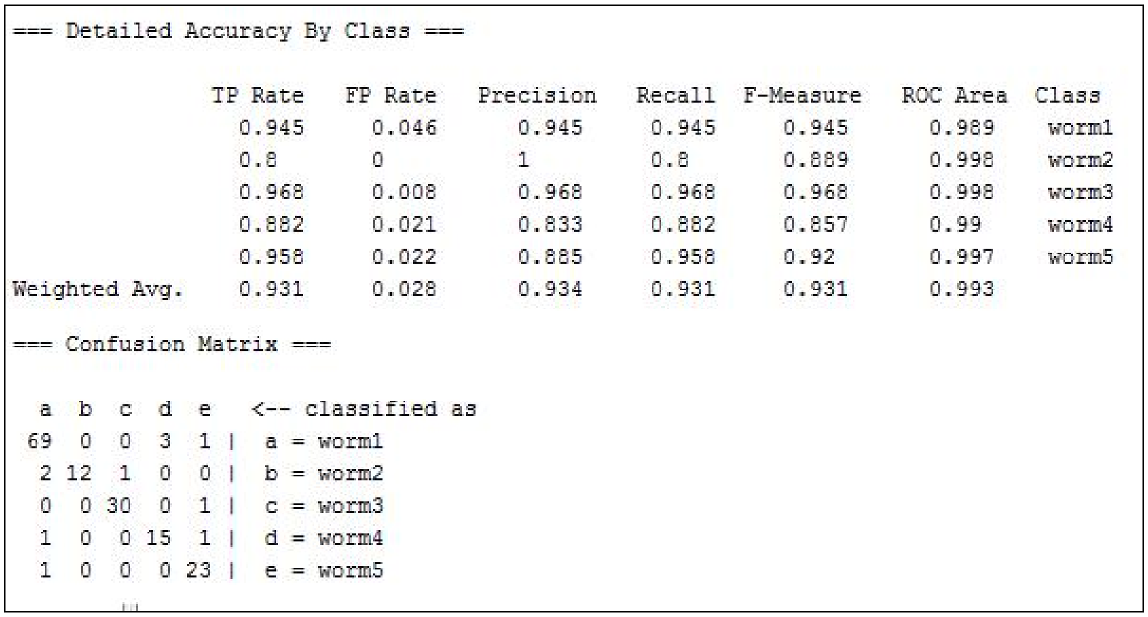
Although the TP rate for worm 2 was only 80%, which was the lowest of all the classes, the FP rate was 0% and the ROC area was 0.998, which indicated an almost perfect performance. The accuracy of each worm class was 95% for worm 1, 98.13% for worm 2, 98.75% for worm 3, 96.88% for worm 4 and 97.5% for worm 5.

However, when the ROC area of worm 3 is looked closely, it has the same value as worm 2 (0.998). As worm 3 has the highest overall accuracy and the lowest FN rate, it can be concluded that worm 3 yielded the highest performance of all the classes. The extracted outputs from the IBk results which consist of ‘detailed accuracy by class’ and ‘confusion matrix’ can be found in Figure 4.21.



```
=== Run information ===

Scheme:       weka.classifiers.lazy.IBk -K 8 -W 0 -I -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""
Relation:     wormclassall_wormed
Instances:    160
Attributes:   7
              Instance_number
              infection
              activation
              propagation
              operating
              payload
              worm
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

IB1 instance-based classifier
using 8 inverse-distance-weighted nearest neighbour(s) for classification


Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         149               93.125  %
Incorrectly Classified Instances        11                6.875  %
Kappa statistic                          0.9033
Mean absolute error                      0.0734
Root mean squared error                  0.1686
Relative absolute error                 25.6542 %
Root relative squared error             44.6619 %
Total Number of Instances              160

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.945     0.046      0.945     0.945     0.945      0.989    worm1
                 0.8       0          1         0.8       0.889      0.998    worm2
                 0.968     0.008      0.968     0.968     0.968      0.998    worm3
                 0.882     0.021      0.833     0.882     0.857      0.99     worm4
                 0.958     0.022      0.885     0.958     0.92       0.997    worm5
Weighted Avg.    0.931     0.028      0.934     0.931     0.931      0.993

=== Confusion Matrix ===

  a  b  c  d  e   <-- classified as
 69  0  0  3  1 |  a = worm1
  2 12  1  0  0 |  b = worm2
  0  0 30  0  1 |  c = worm3
  1  0  0 15  1 |  d = worm4
  1  0  0  0 23 |  e = worm5
```

Figure 4.20. IBk Results.

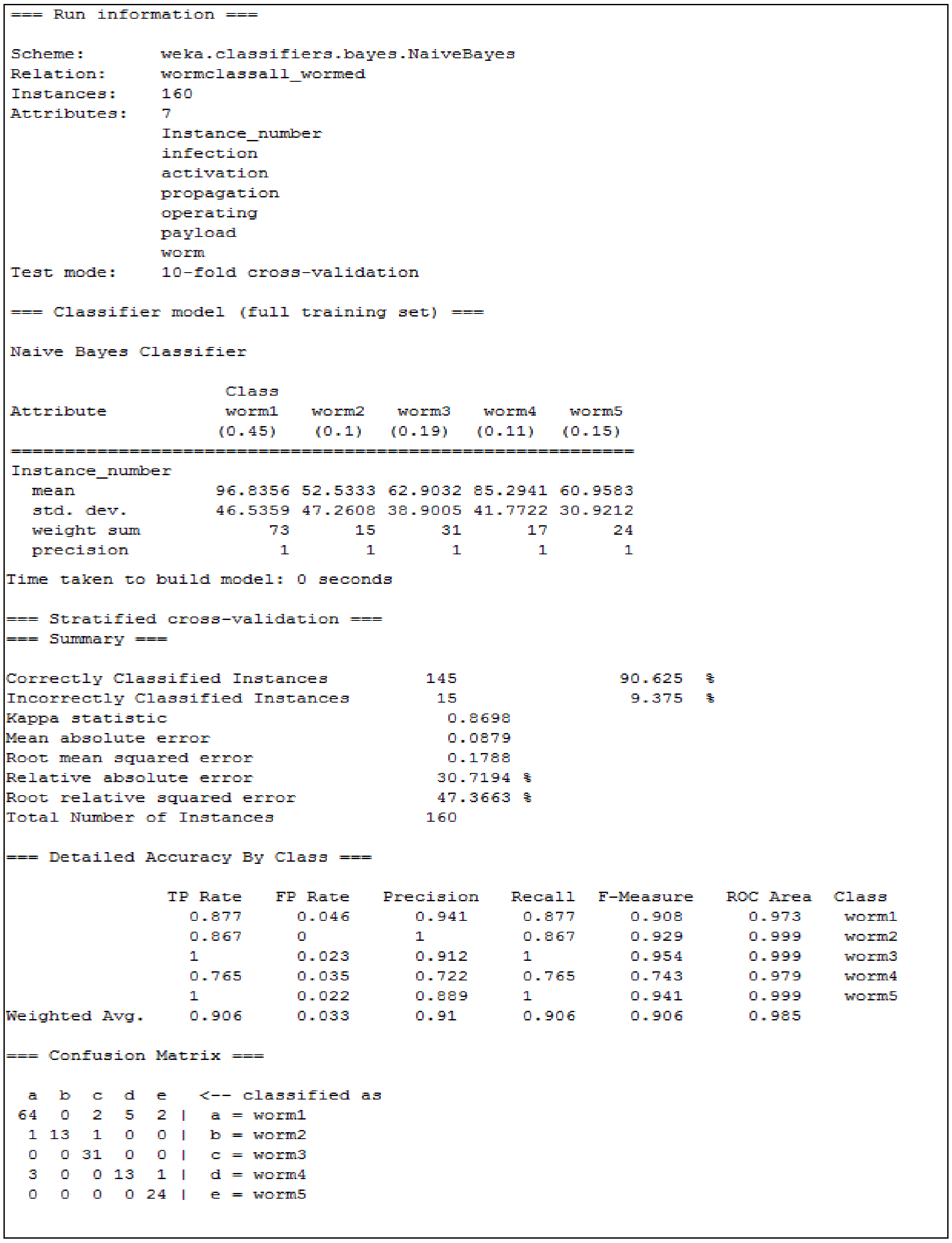


```
=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.945     0.046      0.945     0.945     0.945      0.989    worm1
                 0.8       0          1         0.8       0.889      0.998    worm2
                 0.968     0.008      0.968     0.968     0.968      0.998    worm3
                 0.882     0.021      0.833     0.882     0.857      0.99     worm4
                 0.958     0.022      0.885     0.958     0.92       0.997    worm5
Weighted Avg.    0.931     0.028      0.934     0.931     0.931      0.993

=== Confusion Matrix ===

  a  b  c  d  e   <-- classified as
 69  0  0  3  1 |  a = worm1
  2 12  1  0  0 |  b = worm2
  0  0 30  0  1 |  c = worm3
  1  0  0 15  1 |  d = worm4
  1  0  0  0 23 |  e = worm5
```

Figure 4.21. Extracted Output from IBk Results.

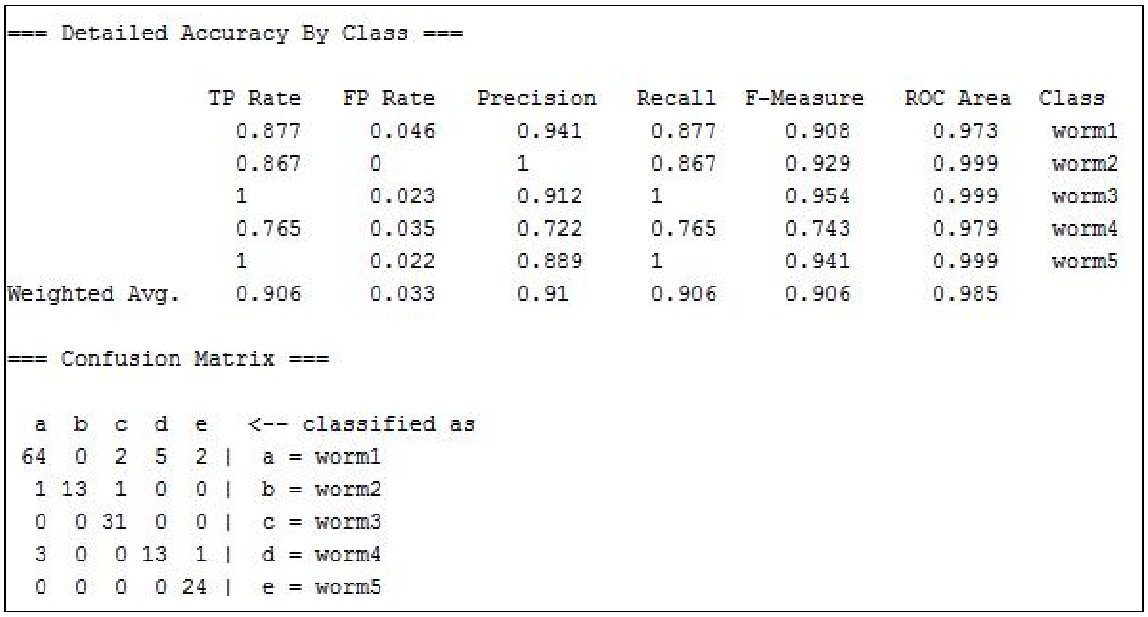
4.4.3.2.4 Naïve Bayes Findings

In this section, the detailed results for the Naïve Bayes algorithm are presented. The overall accuracy for the Naïve Bayes algorithm was 90.63%, while the average FP rate was 3.3% and the average TP rate was 90.6%. Referring to the 'detailed accuracy by class' results in Figure 4.22, the TP rate for worm 1 was 87.7%, worm 2 was 86.7%, worm 3 was 100%, worm 4 was 76.5% and worm 5 was 100%. The FP rate for worm 1 was 4.6%, worm 2 was 0%, worm 3 was 2.3%, worm 4 was 3.5% and worm 5 was 2.2%, while the FN rate for each worm class was 12.33% for worm 1, 13.33% for worm 2, 0% for worm 3 and worm 5 and 23.53% for worm 4. The accuracy for each worm class was 91.88% for worm 1, 98.75% for worm 2, 98.13% for worm 3, 94.38% for worm 4 and 98.13% for worm 5. The extracted outputs from the Naïve Bayes results which consist of ‘detailed accuracy by class’ and ‘confusion matrix’ can be found in Figure 4.23.



```
=== Run information ===

Scheme:       weka.classifiers.bayes.NaiveBayes
Relation:     wormclassall_wormed
Instances:    160
Attributes:   7
              Instance_number
              infection
              activation
              propagation
              operating
              payload
              worm
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

Naive Bayes Classifier

                        Class
Attribute               worm1   worm2   worm3   worm4   worm5
                       (0.45)   (0.1)  (0.19)  (0.11)  (0.15)
==========================================================
Instance_number
  mean                 96.8356 52.5333 62.9032 85.2941 60.9583
  std. dev.            46.5359 47.2608 38.9005 41.7722 30.9212
  weight sum                73      15      31      17      24
  precision                  1       1       1       1       1
Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         145               90.625  %
Incorrectly Classified Instances        15                9.375  %
Kappa statistic                          0.8698
Mean absolute error                      0.0879
Root mean squared error                  0.1788
Relative absolute error                 30.7194 %
Root relative squared error             47.3663 %
Total Number of Instances              160

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.877     0.046      0.941     0.877     0.908      0.973    worm1
                 0.867     0          1         0.867     0.929      0.999    worm2
                 1         0.023      0.912     1         0.954      0.999    worm3
                 0.765     0.035      0.722     0.765     0.743      0.979    worm4
                 1         0.022      0.889     1         0.941      0.999    worm5
Weighted Avg.    0.906     0.033      0.91      0.906     0.906      0.985

=== Confusion Matrix ===

  a  b  c  d  e   <-- classified as
 64  0  2  5  2 |  a = worm1
  1 13  1  0  0 |  b = worm2
  0  0 31  0  0 |  c = worm3
  3  0  0 13  1 |  d = worm4
  0  0  0  0 24 |  e = worm5
```

Figure 4.22. Naïve Bayes Results.



```
=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.877     0.046      0.941     0.877     0.908      0.973    worm1
                 0.867     0          1         0.867     0.929      0.999    worm2
                 1         0.023      0.912     1         0.954      0.999    worm3
                 0.765     0.035      0.722     0.765     0.743      0.979    worm4
                 1         0.022      0.889     1         0.941      0.999    worm5
Weighted Avg.    0.906     0.033      0.91      0.906     0.906      0.985

=== Confusion Matrix ===

  a  b  c  d  e   <-- classified as
 64  0  2  5  2 |  a = worm1
  1 13  1  0  0 |  b = worm2
  0  0 31  0  0 |  c = worm3
  3  0  0 13  1 |  d = worm4
  0  0  0  0 24 |  e = worm5
```

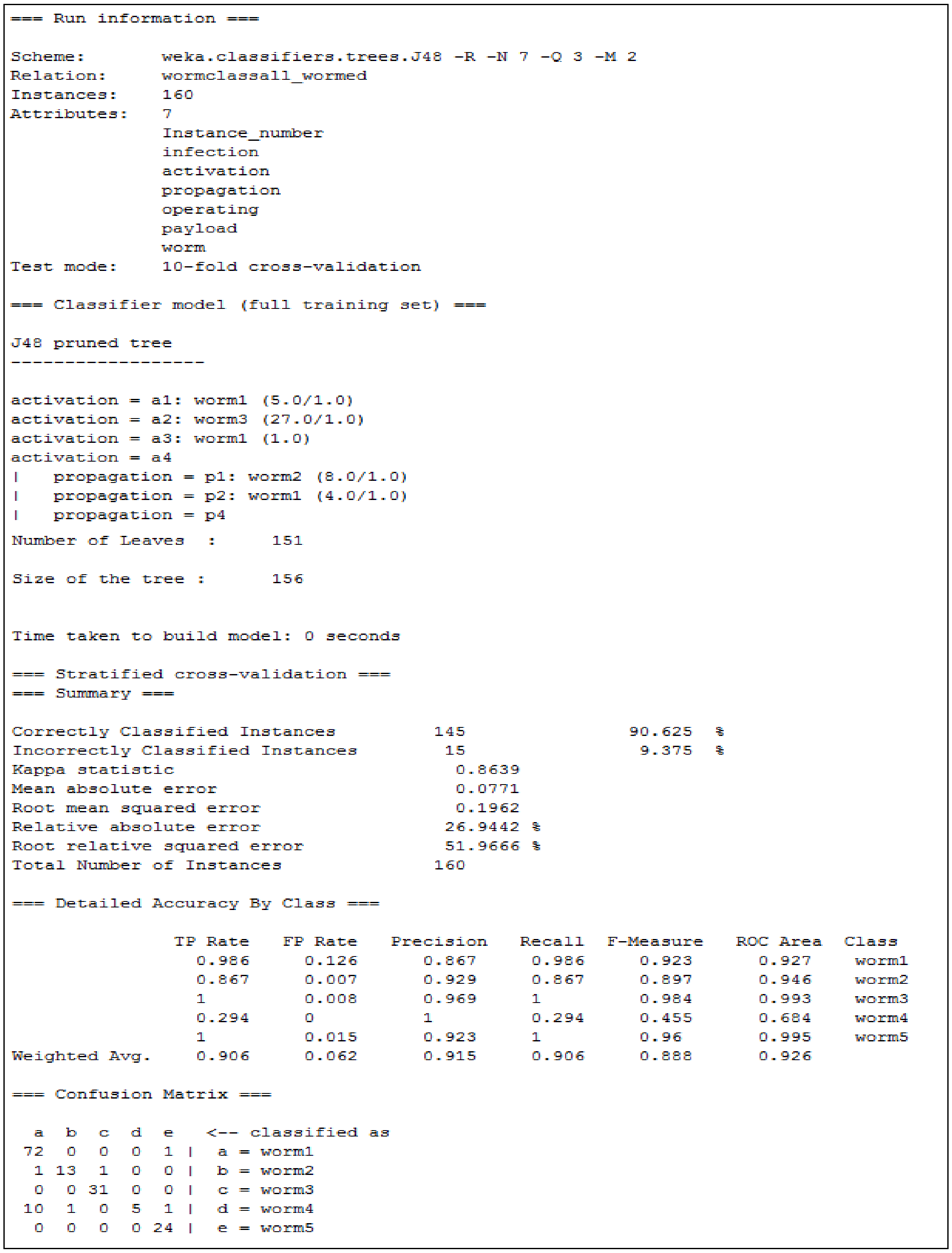
Figure 4.23. Extracted Ouput from Naïve Bayes Results.

When the ROC areas are examined, it can be seen that worms 2, 3 and 5 have the same value (0.999). Although worm 2 had the highest overall accuracy, the FN rate was much higher than worm 3, with a 16.77% difference between them. In order to decide who was the highest performer, in terms of worm detection, (as the two different classes were the same or only slightly different in accuracy) the next performance criteria taken into account was the FN rate: since the implications of a high FN rate are very harmful to a user's computer. Thus, it was concluded that worm 3 yielded the highest performance of all the classes: it had 99.99% of ROC area and 0% of FN rate.

4.4.3.2.5 J48 Findings

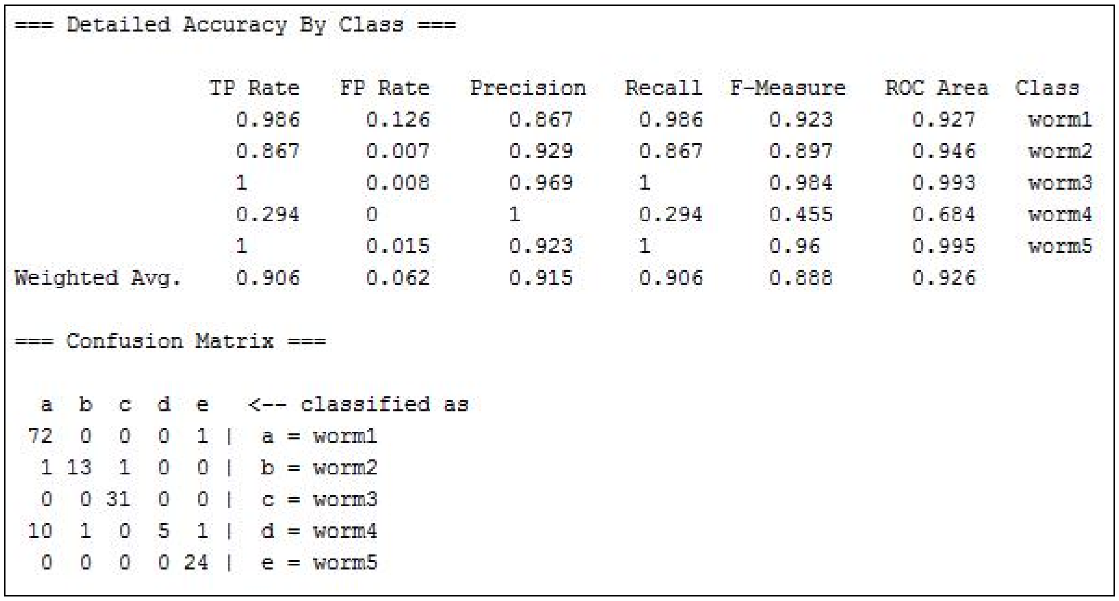
In this section, the detailed results for the J48 algorithm are presented (the J48 algorithm generates the pruned C4.5 Decision Tree and the ID3 descendent). The overall accuracy for the J48 algorithm was 90.63%, the average FP rate was 6.2% and the average TP rate was 90.6%. Referring to the 'detailed accuracy by class' results in Figure 4.24, the TP rate for worm 1 was 98.6%, 86.7% for worm 2, 100% for worm 3, 29.4% for worm 4 and 100% for worm 5. The FP rate for worm 1 was 12.6%, 0.7% for worm 2, 0.8% for worm, 0% for worm 4 and 1.5% for worm 5. The FN rate for each worm class was 1.37% for worm 1, 13.3% for worm 2, 0% for worm 3 and worm 5 and 70.59% for worm 4.

The accuracy for each worm class was 92.5% for worm 1, 98.13% for worm 2, 99.38% for worm 3, 92.5% for worm 4 and 98.75% for worm 5. The extracted outputs from the J48 results which consist of 'detailed accuracy by class' and 'confusion matrix' can be found in Figure 4.25. The TP rate for worm 3 and worm 5 was 100%, while the FP positive rate for worm 3 was 0.8% and 1.5% for worm 5. In order to identify the highest performance between these two classes, the ROC area, accuracy and the FN rate are being referred. The ROC area for worm 5 was slightly higher than worm 2, with a 0.2% difference, whilst the accuracy for worm 3 was higher than worm 5 (0.63%); worm 3 also had 0% of FN rate. Thus, worm 3 yielded the highest performance of the worm classes.



```
=== Run information ===

Scheme:       weka.classifiers.trees.J48 -R -N 7 -Q 3 -M 2
Relation:     wormclassall_wormed
Instances:    160
Attributes:   7
              Instance_number
              infection
              activation
              propagation
              operating
              payload
              worm
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

J48 pruned tree
------------------

activation = a1: worm1 (5.0/1.0)
activation = a2: worm3 (27.0/1.0)
activation = a3: worm1 (1.0)
activation = a4
|   propagation = p1: worm2 (8.0/1.0)
|   propagation = p2: worm1 (4.0/1.0)
|   propagation = p4
Number of Leaves  :     151

Size of the tree :      156


Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         145               90.625  %
Incorrectly Classified Instances        15                9.375  %
Kappa statistic                          0.8639
Mean absolute error                      0.0771
Root mean squared error                  0.1962
Relative absolute error                 26.9442 %
Root relative squared error             51.9666 %
Total Number of Instances              160

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.986     0.126     0.867      0.986     0.923      0.927    worm1
                 0.867     0.007     0.929      0.867     0.897      0.946    worm2
                 1         0.008     0.969      1         0.984      0.993    worm3
                 0.294     0         1          0.294     0.455      0.684    worm4
                 1         0.015     0.923      1         0.96       0.995    worm5
Weighted Avg.    0.906     0.062     0.915      0.906     0.888      0.926

=== Confusion Matrix ===

  a  b  c  d  e   <-- classified as
 72  0  0  0  1 |  a = worm1
  1 13  1  0  0 |  b = worm2
  0  0 31  0  0 |  c = worm3
 10  1  0  5  1 |  d = worm4
  0  0  0  0 24 |  e = worm5
```

Figure 4.24. J48 Results.



```
=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.986     0.126      0.867     0.986     0.923      0.927    worm1
                 0.867     0.007      0.929     0.867     0.897      0.946    worm2
                 1         0.008      0.969     1         0.984      0.993    worm3
                 0.294     0         1          0.294     0.455      0.684    worm4
                 1         0.015      0.923     1         0.96       0.995    worm5
Weighted Avg.    0.906     0.062      0.915     0.906     0.888      0.926

=== Confusion Matrix ===

  a  b  c  d  e   <-- classified as
 72  0  0  0  1 |  a = worm1
  1 13  1  0  0 |  b = worm2
  0  0 31  0  0 |  c = worm3
 10  1  0  5  1 |  d = worm4
  0  0  0  0 24 |  e = worm5
```

Figure 4.25. Extracted Output from J48 Results.

4.5 Comparison with Existing Works

Table 4.2 summarises the results of all the tests conducted and compares them with existing works undertaken by Siddiqui *et al.* (2009) and Dai *et al.* (2009). It was found that their works were similar to this thesis. As seen in Table 4.2, the performance criteria for comparison consists of the TP rate (TPR), overall accuracy (OA), the FP rate (FPR) and the FN rate (FNR). The details of the definition and equation of these performance criteria can be found in Chapter 3, Section 3.2.4.4.

Table 4.2. Experiment Results.

Classifier	STAKCERT Result (%)				Existing Work (%)							
					Dai *et al* (2009)				Siddiqui *et al* (2009)			
	TPR	OA	FPR	FNR	TPR	OA	FPR	FNR	TPR	OA	FPR	FNR
Multilayer Perceptron	**98.88**	**98.75**	**0.2**	**1.45**	NA	NA	NA	NA	NA	NA	NA	NA
SMO	98.1	98.13	0.2	2.63	**93.2**	**91.9**	**9.6**	**6.8**	NA	NA	NA	NA
Naïve Bayes	90.6	90.63	3.3	9.84	NA	NA	NA	NA	NA	NA	NA	NA
IBk	93.1	93.13	2.8	8.93	NA	NA	NA	NA	NA	NA	NA	NA
Decision Tree J48	90.6	90.63	6.2	17.6	93.5	91	12.6	6.5	93.4	90	13.4	6.6
Random Forest	NA	NA	NA	NA	NA	NA	NA	NA	**95.6**	**96**	**3.8**	**4.4**
Bagging	NA	NA	NA	NA	NA	NA	NA	NA	94.3	93.8	6.7	5.7
	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA

** TPR = True Positive Rate (also known as detection accuracy), FPR = False Positive Rate, FNR=False Negative Rate, OA = Overall Accuracy, NA=Not Applicable. Figures in bold show the highest results for each work.*

By referring to Table 4.2, STAKCERT results show that the Multilayer Perceptron algorithm outperformed those of the existing work. Overall accuracy was 98.75%, which is 2.75% higher than Siddiqui's work and 6.85% higher than Dai's work. The STAKCERT TP rate (98.8%) was also higher than in the comparable works and the FN rate (1.45%) was lower. Furthermore, STAKCERT FP rate (0.2%) was also lower.

However in worm detection, a FN rate plays a more important role than a FP rate because a higher FN rate will cause severe damage to a user's computer. When FN rate is higher, this indicates that there are more of the datasets not classified as worms even though actually the datasets are worms. This is the reason why it is important to have a lower FN rate for worm detection testing. Yet in dealing with worms, these four main criteria should always be taken into consideration. A lower FP rate, a lower result for the FN rate and a higher value for the TP rate and overall accuracy are preferable in worm detection. If a result yields the same value for overall accuracy and TP rate and a higher value for the FP rate and different value for the FN rate, the best result should be chosen from the lower FN rate value.

This thesis offers its own significant contribution towards computer security and the novelty of this thesis lies in the method being implemented, where data mining is part of it and the goals achieved by the end of this thesis. This is summarised in Table 4.3. Such improvement implemented methods are the integration of static and dynamic analyses, the statistical analysis and incident response techniques. The work done by Siddique *et al.* (2009) applied the static analysis in their work where the limitation lies when there is a dynamic decision point in the program control flow. Dai *et al.* (2009) overcame this limitation by applying the dynamic analysis. Yet the static or dynamic analysis alone cannot solve the worm detection problem with guarantee. For example, to analyse worm payload, certain worm needs both static and dynamic analyses, so the payload can be monitored and executed. Therefore, STAKCERT model has combined both static and dynamic analyses to provide an improved detection result as shown in Table 4.2.

Prior to the results retrieved (Table 4.2), the standard operating procedures using incident response techniques were used, in order to conduct static and dynamic analysis on the worm. In contrast with the existing works where they do not integrate the standard operating procedures using the incident response technique, their methods can be arguable. The standard operating procedures ensure all the related procedures are followed accordingly before and during the worm analysis and all related procedures documented.

Table 4.3 Comparison with Existing Works for Worm Detection.

	STAKCERT	Existing Works	
		Dai *et al* (2009)	Siddique *et al* (2009)
Method of analysis	1) Involves dynamic analysis. 2) Involves static analysis. 3) Integrates standard operating procedures using the incident response technique. 4) Involves statistical analysis: Independent testing (Chi-square, symmetric measure and frequency analysis). 5) Applies data mining as part of STAKCERT KDD processes to model building.	1) Involves dynamic analysis. 2) Does not involve static analysis. 3) Does not integrate standard operating procedures using the incident response technique. 4) Involves statistical analysis: Frequency analysis. 5) Applies data mining as a complete process from data preparation to model building.	1) Does not involve dynamic analysis. 2) Involves static analysis. 3) Does not integrate standard operating procedures using the incident response technique. 4) Involves statistical analysis: Independent testing (Chi-square and frequency analysis). 5) Applies data mining as a complete process from data preparation to model building.

Once the static and dynamic analyses were completed, a STAKCERT worm classification was formed. The relationships between the main features within the STAKCERT worm classification were then verified by undertaking statistical analysis, in order to show the relationship amongst these features. The statistical analysis consists of Chi-square, symmetric measure and frequency analysis. Such features were later used as the input for the data mining analysis, which resulted in a higher overall performance. As for Dai *et al* (2009) they applied frequency analysis and Siddique *et al* (2009) applied frequency analysis and Chi-square.

For this thesis, data mining is a part of the STAKCERT KDD processes (refer Figure 3.5) used to optimise worm detection accuracy. In this thesis, the static analysis, dynamic analysis, standard operating procedures of incident response, Chi-square, symmetric measure and frequency analysis are part of the whole STAKCERT KDD processes. The STAKCERT KDD processes are used to build the STAKCERT model. In contrast with Dai *et al.* (2009) and Siddique *et al.* (2009) works, they used data mining as a process to form their model. The better result accuracy achieved and presented in Table 4.2 is therefore as a result of the STAKCERT KDD processes.

In conclusion, this thesis results yielded a better performance than comparable, existing work which could be due to the improvement made by applying both static and dynamic analyses and statistical analysis(i.e: Chi-square, symmetric measure and frequency analysis) and by integrating the standard operating procedures using an incident response technique. Such results were used as the input in triggering the apoptosis process, which is discussed in the next chapter.

4.6 Limitations

In this thesis, a performance comparison for the different learning algorithms that were applied to the datasets is conducted and the only apparent drawback of the MLP algorithm is that it requires more training time than other algorithms. In addition, this thesis may be improved by considering different types of malicious code, such as spyware, Trojan horse and botnet. Apart from that, the integration of dynamic and static analyses may require more investigation and refinement to produce a better result for worm detection, which is to be explored in the future. Furthermore, an expansion of the different types of datasets would improve the robustness of the STAKCERT model, although a few modifications would have to be implemented under the pre-processing procedures.

4.7 Summary

In this section, the STAKCERT worm classification and the STAKCERT relational model are proposed, which are both part of the STAKCERT model for worm detection. Experimental results indicate that the proposed model can detect worms, with as high as a 98.75% overall accuracy rate and as low as a 0.2% FP rate and a 1.45% FN rate. The comparison of STAKCERT model with existing work showed that STAKCERT model for worm detection resulted in improved performance.

CHAPTER 5

MODELLING STAKCERT FOR WORM RESPONSE

Chapter 5 explains the details of the STAKCERT model for worm response. This contribution relates to how the end user responds towards a worm incident where apoptosis is part of the response. Apoptosis, also known as cell-programmed death, is a concept borrowed from the human immune system (HIS). Once the user's computer detects any indication of being infected severely by a worm, apoptosis is triggered, which isolates the infected computer from any network. In order to trigger apoptosis, the weight and the severity value of the worm play important roles, since these two factors help to decide either apoptosis should be triggered or not. An in-depth study was carried out by implementing security metrics in identifying the weight and severity of the infection, which resulted in new STAKCERT apoptosis algorithm for detecting worms. Based on the experimental results, the assigned rate of severity was 100% accurate. Furthermore, the STAKCERT model was simulated with the eradication solutions, which yielded an overall accuracy rate of 98.08% and F-measure rate of 100%. The performance criteria results indicated that the STAKCERT model was an efficient worm response model.

5.1 Introduction

Over the last few years, there has been increasing interest in studying the human immune system (HIS). Computer scientists, engineers, mathematicians, philosophers and other researchers are particularly interested in HIS' capabilities, the complexity of which is comparable to the brain. HIS is not new and much research has been published since 1996 such as by Hunt and Cooke (1996), Dasgupta (1997), Dasgupta (1999) and Hofmeyr and Forrest (1999). Apoptosis is part of these studies.

In the human body, apoptosis also known as cell-programmed death is used to destroy cells infected with a virus, cells with DNA damage, and some cancerous cells, which may be a threat to the organism. The main benefit of apoptosis is that cells can be disposed of without causing harm or stress to other cells in the same part of the body. Apoptosis is a process that prevents the virus in the infected cell from spreading to other parts of the body which could cause a lot of trouble to the overall system (Raff 1998). Chapter 2 provides details of apoptosis and compares it with worm security problems.

From a worm response perspective, apoptosis is implemented to avoid the worm propagating to other computers in the same network or via the Internet. Prior to apoptosis, there are several factors which should be taken into consideration. In Chapter 4, the detection of worms was based on the five main characteristics of a worm, which were based on the STAKCERT worm classification and the STAKCERT relational model. Furthermore, for apoptosis, these five main characteristics of a worm are further refined and reused by assigning it with a weight and severity value, based on security metrics method. Security metrics is explained in Chapter 3 (section 3.2.4.3). Based on this thesis analysis and experimentation with regard to the security metrics, the data criticality, infrastructure availability and loss of productivity were used as the basis for assigning a weight and severity value. Table 3.5 in Chapter 3 shows the security metrics processes already mapped into the STAKCERT model. As a result, the STAKCERT worm apoptosis algorithm was formed. Section 5.3 explains this in detail. The simplified flowchart for the weight and assigned severity values are shown in Figure 5.1.

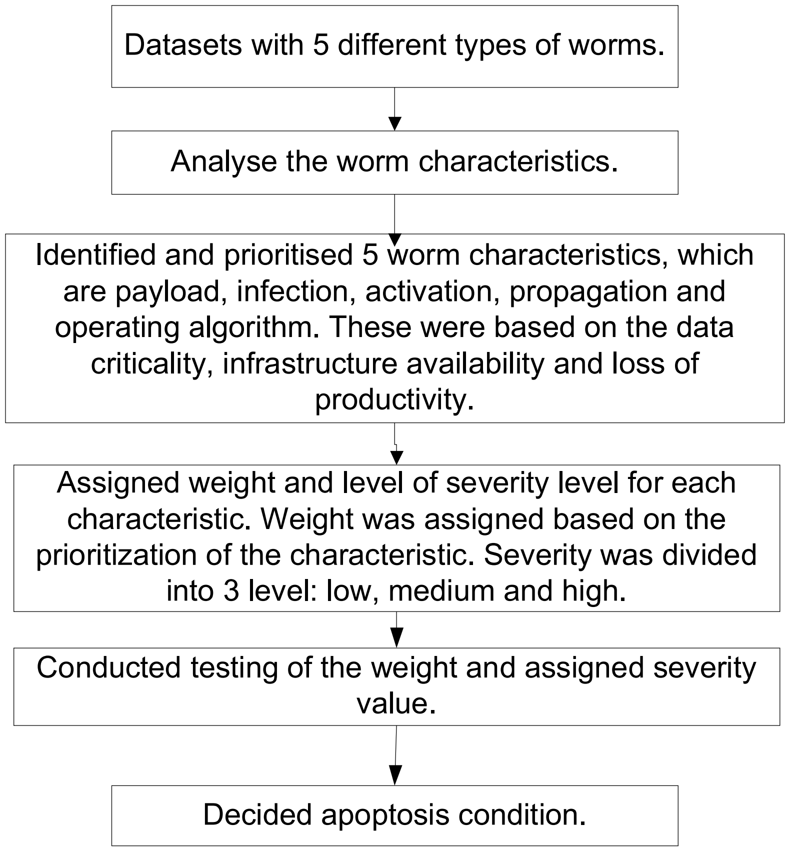

Datasets with 5 different types of worms.
Analyse the worm characteristics.
Identified and prioritised 5 worm characteristics, which are payload, infection, activation, propagation and operating algorithm. These were based on the data criticality, infrastructure availability and loss of productivity.
Assigned weight and level of severity level for each characteristic. Weight was assigned based on the prioritization of the characteristic. Severity was divided into 3 level: low, medium and high.
Conducted testing of the weight and assigned severity value.
Decided apoptosis condition.

Figure 5.1. Weight and Severity Assignment Flowchart.

5.2 Related Works

Apoptosis provides a lot of scope for exploring its implementation or integration in the computer security field. Prior to the introduction to the STAKCERT model for worm response, a thorough study of the existing literature on apoptosis was undertaken as already discussed and outlined in Chapter 2. The challenge, which should be considered thoroughly from all of these previous works, was the method of assigning apoptosis and the scope of its implementation. These are still lacking in handling the response to a worm incident. For the past few years, much research is focusing on worm detection though worm response has the same important role in confronting worm attacks. It is suggested here that results may be improved by considering the weight and severity value, which triggers apoptosis and focuses on responding to a worm incident. This has been taken into consideration when developing the STAKCERT model. In the next section, this thesis explains in detail how weight and severity are integrated into the STAKCERT model. The security metrics and frequency analysis were used to retrieve the rank and the value of the weights and the severity.

Furthermore, in order to test the effectiveness of the STAKCERT model for worm response, a comparison of the work with research conducted by Kim *et al.* (2010) and Liu *et al.* (2010) was undertaken. Kim *et al.* (2010) implemented a system called DSS, which applied a collaborative response, whilst Liu *et al.* (2010) implemented a system using an ontological approach. According to Liu *et al.* (2010), ontology is a term borrowed from philosophy that is used to provide formal specification in a domain, where the concepts and relationships that exist between entities are part of it. As for Liu *et al.* (2010), ontology is used to represent the security incident based on incident response to retrieve the best match incident case. Both of them had the same objective, which was to respond to the incident. The improvement made in the STAKCERT model compared to these two works was to add one further new step. This was applying apoptosis during the response process and the scope of implementation, where the STAKCERT model was dedicated, especially, for detecting and responding to a worm.

5.3 STAKCERT Model for Worm Response

The following are the details of the formation of the STAKCERT model for responding to a worm. It consists of the algorithm and rules for worm apoptosis, weight and severity.

5.3.1 STAKCERT Worm Apoptosis Algorithm

An overview of the pseudocodes to generate the STAKCERT worm apoptosis algorithm is simplified as the following:

```
Given:

- Set security metrics.
- Set worm attributes: {payload, infection, activation, operating algorithm and
  propagation}.
- Set frequency analysis.

Output:
- Weight ranks.
- Severity ranks.
- Weight values.
- Severity values.
- Triggers or halts Apoptosis.

Algorithms:
    1) Apply security metrics to worm attributes.
           a. Go to Weight_cases to determine the weight ranks.
           b. Go to Severity_cases to determine the severity ranks.
    2) Apply frequency analysis to worm attributes.
           a. Go to Frequency_cases to compute the weight and severity values.
    3) Apply apoptosis to Severity_cases.
           a. Go to Apoptosis_cases to trigger the apoptosis.
```

Figure 5.2. An Overview of STAKCERT Worm Apoptosis Algorithm.

A detail of the pseudocodes used to generate the STAKCERT worm apoptosis algorithm are as follows: Weight_cases (refer Figure 5.3); Severity_cases and Apoptosis_cases(which both are combined together in pseudocodes called Severity_cases and Apoptosis_cases – refer Figure 5.4); and Frequency_cases (refer Figure 5.5). All of these pseudocodes explain how the weight and severity was assigned for each attribute of the worm. The attributes were the payload, infection, activation, propagation and operating algorithm. Covering algorithm, also known as the separate-and-conquer algorithm is used to form the STAKCERT worm apoptosis algorithm. Indeed rules were formed as part of the STAKCERT worm apoptosis algorithm. Based on this covering algorithm, there is a rule for the attributes in each stage. It was based on the PRISM method for constructing rules and generated only correct or perfect rules with 100% accuracy (Witten and Frank 2005). The accuracy formula uses p/t where p represents the positive examples of the class and t represents the total of the datasets. The covering algorithm was applied to generate the rules in the STAKCERT worm apoptosis algorithm.

Nevertheless, these algorithms lead to the creation of the STAKCERT rules for weight, severity and apoptosis which can be referred in Appendix C. The Weight_cases and Severity_cases pseudocodes in Figure 5.3 and Figure 5.4 accordingly, show how all of the worm attributes consisting of infection, activation, payload, operating algorithm and propagation were assigned with either a low, medium or high weight, which later resulted in the assignment of severity ranks. For example if a payload with security metrics is high either singly or in combination with other attributes; then the weight is high. While for severity assignment the example is, if payload or activation is high either singly and the weight combination of the propagation, infection and operating algorithm is high, medium or low; then the severity is high.

Initially the following worm characteristics weight are being assigned as high, medium or low based on security metrics (i.e data criticality,infrastructure availability and loss of productivity). The details of how worm weight categorisation assigned can be referred in Appendix C. Basically the rule is presented in *IF-THEN-ELSE* form. The rules are expressed in the form of:

If (Attribute-1, value -1) and (attribute -2, value -2) and….

and (attribute –n, value –n) then (decision, value)

For example :

1) Firstly, define each weight for each of the worm characteristics. For example worm X, has the following features:

Infection	vulnerability	Referring to the weight rules in assigning weight in Appendix C, this characteristic is categorised as *High. (Based on rule no 29)*
Payload	Mass mailing and forward user's info to the attacker	Referring to the rules for weight assignment in Appendix C, these characteristics are categorised as *High. (Based on rule no 13 and 10)*
Activation	Self activation	Referring to the weight rules in assigning weight in Appendix C, this characteristic is categorised as *High.* *(Based on rule no 34)*
Propagation	None	Referring to the weight rules in assigning weight in Appendix C, this characteristic is categorised as *Low.* *(Based on rule no 43)*
Operating Algorithm	TSR	Referring to the weight rules in assigning weight in Appendix C, this characteristic is categorised as *Medium.* *(Based on rule no 46)*

2) Based on the weight from above table, the severity is being assigned.

Infection	Payload	Activation	Operating Algorithm	Propagation	Severity
High	High	High	Medium	Low	Referring to the rules for severity assignment in Appendix C, this characteristic is categorised as *High.* *(Based on rule no 2)*

3) Then apoptosis is being assigned based on the severity weight.

Infection	Payload	Activation	Operating Algorithm	Propagation	Severity	Apoptosis
High	High	High	Medium	Low	High	Referring to the rules for apoptosis assignment in Appendix C, this characteristic is categorised as *High.* *(Based on rule no 2)*

The above worm X characteristics rule is based on severity and apoptosis rules where:

If it involves the combination of rule 1* **and** the weight combination of the *propagation, activation* and *operating algorithm* is high, medium or low, **then** the severity is *high* **and** triggers *apoptosis.*

rule 1 is from severity and apoptosis assignment in Appendix C. (rule 1:* **If the weight for the *payload* and *infection* is high, **then** the severity is *high* **and** triggers *apoptosis.)*

Weight_cases Pseudocodes

```
Given:
- Set characteristics value : {low, medium, high}
- Set HighFlag = 0
- Set MediumFlag = 0
- Dataset A
Output:
- Weight rank of Dataset B
Algorithms:
While (case ≤ 160)
{
  -  get the worm attributes

    While ( worm_attributes != null)
    {
         While (Dataset A != empty)
          {
            - determine characteristic value for each type of worm attributes from the Dataset A
            - Dataset B = Dataset A (worm_attributes[case,type])
                    If (characteristic_value = high)
                        {
                         HighFlag = 1
                      break
                        }
                    If (characteristic_value = medium)
                        MediumFlag = 1
           }
                    If (HighFlag = 1)
                      Weight_rank = high.
                    else if (MediumFlag = 1)
                      Weight_rank = medium.
                   else
                       Weight_rank = low
            -  get the next worm attributes from Dataset A
         }
}
```

Figure 5.3. Weight Cases Pseudocodes.

Severity_cases and Apoptosis_cases Pseudocodes

Input:
- Weight rank of Dataset B

Output:
- Severity rank of Dataset B
- Apoptosis triggers or halts

Algortihms:

```
While  ( case ≤ 160 cases )
{
        If Dataset B (case[payload]) || Dataset B (case[infection])  ==
          High
          {
              SeverityRank [case] = High
              Apoptosis = Triggers
            -  disconnect network and notify user
           }
         Else if Dataset B (case[payload]) || Dataset B (case[infection])  == Medium
          {
              SeverityRank [case] = Medium
              Apoptosis = Halts
              -  network connected and notify user
          }
         Else
          {
              SeverityRank [case] = Low
              Apoptosis = Halts
              - network connected and does not notify user
        - get the next case from Dataset B
}
```

Figure 5.4. Severity Cases and Apoptosis Cases Pseudocodes.

Frequency_cases Pseudocodes

Input:
- Weight rank of Dataset B
- Severity rank of Dataset B

Output:
- Weight values
- Severity values

Algorithms:

```
While  ( case ≤ 160 cases )
{
         - get the weight rank and severity rank for worm attributes from
           Dataset B
          - compute the weight rank and severity rank
          - get the total for weight rank and severity rank

- get the next worm attributes and case from Dataset B

}
```

Figure 5.5. Frequency Cases Pseudocodes.

The Severity_cases and Apoptosis_cases pseudocodes were generated to decide (based on the assigned severity) whether or not apoptosis should be carried out. The apoptosis is fundamentally is a binary, which is either to disconnect or to remain connected to the network. In this thesis, three level of severity categorization is used which are high, medium and low. The main reason of using three level of severity is because each severity level has it owns respond method. If the severity value is high, then apoptosis is triggered, the user is notified and the network is disconnected. When the severity is medium, the apoptosis is halted, the user is notified, and the network is still connected. If the severity is low, apoptosis is halted, the user is not notified, and the network is still connected. In practise, alternatively a binary classifier could be used which is either to connect or disconnect the network instead of using three level of severity categorisation, as being proposed in this thesis.

As for Frequency_cases pseudocodes, these were generated to get the worm attributes ranking and to retrieve exact value for each worm attribute which later was used for the model simulation purpose in section 5.4.2.

The rationale for selecting the covering algorithm for the formation of the STAKCERT worm apoptosis algorithm was its capabilities to develop an algorithm based on the datasets. These were provided by separating them from the datasets already created by the rule. Then, the rule developing process continued on those datasets that remained. The algorithm used in this covering algorithm, increased the effectiveness of the rules since each rule was revised until it became ideal, and the rules developed could be executed independent of order. The only limitation was the need for a revision when conflicting rules occurred.

Based on the STAKCERT worm apoptosis algorithm, there were 66 rules generated. These consisted of 48 rules for weight assignment and 18 rules for severity and apoptosis. Moreover, the structural pattern from these rules can be generated from the weight rules and the severity and apoptosis rules. These structural pattern rules are simplified in a table and the details of these rules can be seen in Appendix C.

5.3.2 Weight and Severity

Studies and experiments on weight and severity are very important in triggering apoptosis. Earlier, in section 5.3.1, the algorithms on how to assign the weight and severity were explained in detail. Based on studies of previous works, there was no standard in assigning weight. Therefore, the data criticality, infrastructure availability and loss of productivity are used, which was part of the security metrics, as a basis and guide for assigning weight and severity. Moreover, this thesis adopted a novel approach to the assignment of weight and severity, which resulted in apoptosis. This makes the STAKCERT model for worm response unique.

Furthermore, to retrieve the exact number of values for each of the worm's attributes, relative frequency is used. The relative frequency for each attribute was based on the STAKCERT worm apoptosis algorithm, and further tested with different algorithms to identify the best overall accuracy value. The equation used for relative frequency is shown in equation 12.

$$rf_n(E) = \frac{r}{n} \tag{12}$$

where

rf_n = relative frequency

E = number of events

n = total number of experiments conducted

r = number of times an event occurs

Relative frequency is another term for proportion. It is the value calculated by dividing the number of times an event occurs by the total number of times an experiment is carried out. Since the cases involved a long run relative frequency, probability was seen as the best way to calculate the weight. It was in the range of 0 to 1. The equation is simplified in equation 13.

$$P(E) = \lim_{n \to \infty} rf_n(E) \tag{13}$$

where,

$P(E) =$ *number of outcomes corresponding to event E / total number of outcomes*

rf_n = relative frequency

Based on the frequency analysis, the worm's attributes are ranked. The next section, 5.4.1, details the frequency analysis results and worm attribute rankings.

5.4 Experimental Results on VX Heavens Datasets

This section presents the results for weight and severity, based on the frequency analysis. This section also presents the simulation results for the STAKCERT model for responding to a worm.

5.4.1 Weight and Severity Results

The frequency analysis was conducted to support the fact that the weight, severity and apoptosis algorithm, formed under section 5.3.1 and based on security metrics, were effective. Furthermore, based on the frequency analysis testing results in Table 5.1, the ranking of worm attributes was identified as follows:

(1)Payload ; (2) Infection ; (3) Activation; (4) Propagation and (5) Operating algorithm

Table 5.1. Frequency Analysis Results.

	Payload		Infection		Activation		Propagation		Operating Algorithm		Severity	
High	150	0.938	129	0.806	121	0.756	22	0.138	8	0.05	430	0.538
Medium	9	0.056	26	0.163	33	0.206	0	0	152	0.95	220	0.275
Low	1	0.006	5	0.031	6	0.038	138	0.863	0	0	150	0.188
Total	160	1	160	1	160	1	160	1	160	1	800	1

Table 5.2. Severity Results using Different Algorithms.

Classifier	Multilayer Perceptron			SMO			Naïve Bayes			J48			IBk		
Severity in %	H	M	L	H	M	L	H	M	L	H	M	L	H	M	L
TPR	100	83.3	0	100	100	0	99.4	100	0	98.7	83.3	0	100	83.3	0
FPR	16.7	0	0	0	0	0	0	0.6	0	16.7	1.3	0	16.7	0	0
FNR	0	16.7	0	0	0	0	0.6	0	0	1.299	16.7	0	0	16.7	0
OA	***99.38***	***99.38***	***0***	***100***	***100***	***0***	***99.38***	***99.38***	***0***	***98.13***	***98.13***	***0***	***99.38***	***99.38***	***0***

**TPR=true positive rate, FPR=false positive rate, FNR=false negative rate, OA = overall accuracy, H= high, M= medium, L=low.*

160 cases were tested from the datasets. Then, the accuracy of each case with an assigned severity value was further tested using different data mining algorithms by means of WEKA. Table 5.2 presents the results.

With regard to Table 5.2, five different algorithms were tested. These were the Multilayer Perceptron, SMO, Naïve Bayes, J48 and IBk. The results of each algorithm are detailed in Appendix E. The equations detailing the overall accuracy, FP rate, FN rate and TP rate and the details of each algorithm conducted, are provided in Section 3.2.4.4. The objective of this testing was to identify the overall accuracy for the assigned class of severity. Each of the worm's attributes, which were payload, infection, activation, propagation and operating algorithm, were assigned with either a high, medium or low weight value. Based on the testing conducted, the average overall accuracy for each algorithm was more than 98.13%. SMO had the highest overall accuracy of 100%, followed by 100% of TP rate and 0% of both the FP rate and FN rate of severity assigned. The overall accuracy rate of Multilayer Perceptron, Naïve Bayes and IBk yielded was 99.38%, and the overall accuracy rate of J48 was 98.13%.

These excellent results, as noted in Table 5.2, were based on the proper and effective algorithm implemented prior to this testing. This included the assignment of the weight, severity and apoptosis algorithms and the significant STAKCERT worm classification and relational model formed earlier.

5.4.2 STAKCERT Model Simulation for Apoptosis Results

By the end of this research, it was this research aims that the developed STAKCERT model should be implemented in real time for worm detection and response incident software. Therefore, a simulation of the model was carried out using WEKA. At this point, all the attributes which had been tested in Chapter 4 together with the weight, severity and eradication solution, were simulated to test the rate of accuracy, and to identify the Precision, Recall and F-measure results. Since the aim for this simulation was to test apoptosis, only 156 relevant high severity cases which would trigger apoptosis, were tested. Table 5.3 summarises the results of this testing. This simulation was simulated to know whether or not the retrieved eradication solution was relevant. The most important performance elements of the STAKCERT model were based on the Precision, Recall and F-measure results. The higher Recall value suggested that the relevant solution was returned more quickly, and that the higher Precision value meant that the returned solution was more relevant. Moreover, the model's performance could be measured in terms of a single measure of performance by using the F-measure, which was a combination of the Recall and Precision values. Section 3.2.4.4 details the Precision, Recall and F-measure equations.

With regard to Table 5.3, the Multilayer Perceptron algorithm yielded the highest overall performance, with an overall accuracy of 98.08% and an average F-measure of 100%. The averages of both the Precision and Recall values were 100%. This showed that 100% of the returned solutions were the quickest and were rightly relevant. Although the overall accuracy was not 100%, there is always room to improve the accuracy of the results produced. All of these results showed good indication and promise for the future and consequently, the STAKCERT model could be implemented in real time worm detection and response incident software.

The second highest overall performance was by the SMO algorithm with an overall accuracy of 96.79%, while the F-measure averaged 97%, the Precision averaged 100% and the Recall averaged 94.1%. Even though the other results were below those produced by the Multilayer Perceptron result, the 100% Precision value indicates its abilities to retrieve the most relevant solution.

The overall accuracy of the IBk algorithm was 96.15%, with the F-measure averaging 94.1% and the Precision and Recall averaging 94.1%. The results showed that it had the ability to return and retrieve 94.1% from the relevant solution, though it was not as high as the result produced by the Multilayer Perceptron.

As for J48, it had a 100% Precision value but a lower Recall value of 88.2%, which resulted in an F-measure value of 93.8%. The Recall value indicated a lower ability to retrieve the solution, but the overall accuracy was still 94.23%, which could be considered a good result. Lastly was the Naïve Bayes with an overall accuracy of 92.31% and a 100% Recall value. However, the Precision was only 81% which resulted in the value of the F-measure being 89.5%. If this algorithm were to be implemented, improvement has to be made prior to this.

In conclusion, a comparison of the five different algorithms tested showed that the Multilayer Perceptron yielded the highest overall performance criteria result. This indicated that the STAKCERT model is an effective model. With the integration of the Multilayer Perceptron and STAKCERT model, there is no doubt that its implementation for future worm detection and response incident software would provide promising results.

Table 5.3. The Simulation Results for STAKCERT Model Worm Simulation.

Algorithm	Multilayer Perceptron				SMO				IBk				Naïve Bayes				J48			
Performance criteria	FM	PR	RC	OA	FM	PR	RC	OA	FM	PR	RC	OA	FM	PR	RC	OA	FM	PR	RC	OA
Average result in %	100	100	100	98.08	97	100	94.1	96.79	94.1	94.1	94.1	96.15	89.5	81	100	92.31	93.8	100	88.2	94.23

**FM= F-measure, PR=Precision, RC= Recall, and OA=overall accuracy*

5.5 Comparison with Existing Works

Even though there is no specific measurement that can be compared for worm response, it is possible to make a comparison with other related works to create a point of reference. In terms of an incident response perspective, this thesis has improved works done by Kim *et al.* (2010) and Liu *et al.* (2010) by adding one further step to the worm response and refining procedures prior to worm response. This is done by applying security metrics method, standard operating procedures in incident response before the worm response and apoptosis during the worm response process. A comparison with existing work is summarised in Table 5.4.

Table 5.4. Comparison with Existing Works for Worm Response

	STAKCERT	Existing Works	
		Kim *et al* (2010)	Liu *et al* (2010)
Methods of responding to the incident.	1) STAKCERT model consists of STAKCERT KDD processes to detect and respond to the incident. SOP in IR is part of the STAKCERT KDD processes (refer Figure 3.5).	1) Applies DSS framework which is the combination of Recency, Frequency, Monetary (RFM) analysis methodology and CBR to detect and respond to the incident.	1) Applies ontology and CBR to detect and respond to the incident.
	2) Applies security metrics for weight and severity assignment prior to the incident response which leads to the formation of a new STAKCERT worm apoptosis algorithm.	2) Does not apply security metrics and does not have weight and severity assignment prior responding to an incident.	2) Does not apply security metrics and does not have weight and severity assignment prior responding to an incident.
	3) Applies apoptosis to stop worm from further propagation.	3) Does not have specific method to stop worm from further propagation.	3) Does not have specific method to stop worm from further propagation.

Kim *et al.* (2010) developed a DSS framework which is based on Recency, Frequency, Monetary (RFM) analysis methodology and case-based reasoning (CBR) and Liu *et al.* (2010) applied ontology and CBR to detect and respond to the incident. These works could greatly improve matters by detailing the required procedures for handling a worm incident. This is one of the precepts of the formation of the STAKCERT model for worm detection and response, of which incident response is a part. The result of worm detection accuracy in Table 4.3 has indicated the effectiveness of applying standard operating procedures (SOP) in incident response (IR).

Furthermore, to respond to a worm incident in deciding whether or not the incident is severe enough, a method called security metrics is used to help in quantifying, classifying and measuring information in security operations. In Table 3.5 is a summarisation on how security metrics was being applied in this thesis. The security metrics helps us to assign the weight and severity ranks which is either low, medium or high based on data criticality, infrastructure availability and loss of productivity. Later, the apoptosis is triggered based on the weight and severity rank. If the weight and severity rank are high, apoptosis is triggered and the network will be disconnected to avoid the worm from spreading further. Based on the experimental results conducted, the assigned rate of severity was 100% accurate. The STAKCERT worm apoptosis algorithm was introduced, which explained in detail how to assign the weight and severity values to trigger apoptosis by using the security metrics approach. The security metrics and SOP in IR are parts of the STAKCERT KDD processes. To establish if the stated methods applied above were working effectively, the STAKCERT model was simulated with the eradication solutions and yielded an overall accuracy rate of 98.08% and F-measure rate of 100%. These results indicate that the STAKCERT model is an effective worm response model.

In addition, the existing work by Kim *et al.* (2010) and Dai *et al.* (2010) looks at detecting and responding to incidents. This may need further work into detail their detection and response solutions. In contrast, the STAKCERT model was built specifically to detect and respond to worm incidents. A thorough study and experiments carried out on worm incidents leads this thesis to the development of the STAKCERT worm apoptosis algorithm.

In conclusion, the STAKCERT model has a promising future to be implemented as worm detection and response software based on the methods introduced which consist of SOP in IR, security metrics and apoptosis. This is suggested here as future work.

5.6 Limitations

The test conducted were based on a simulation using the WEKA software. If the STAKCERT model was to be implemented in real time, the retrieval method should be improved for a better accuracy. In addition, the proposed STAKCERT model is based on a worm associated with Windows applications. Future work could expand scope of this research to provide greater opportunity to explore the different types of malicious code and the use of this model on different platforms.

5.7 Summary

Based on the results of the experiments and testing conducted, the STAKCERT model for worm response successfully achieved its objective with an overall accuracy rate of 96.08% and 100% F-measure value. Prior to that, the weight and the severity assigned to the worm characteristics were tested and showed an overall accuracy rate of 100%. Moreover, the novelty of this thesis in worm response lies in the implementation of apoptosis. As part of this, studies and experiments were conducted into the assigned weights and levels of severity in order to trigger apoptosis. Furthermore, the STAKCERT worm apoptosis algorithm was developed by integrating the weight and level of severity. Some indication of how this work can be developed and improved is discussed in Chapter 6.

CHAPTER 6

CONCLUSION AND FUTURE WORK

The main contribution of this thesis is the development of a new model called STAKCERT for worm detection and response. The strength of this model lies in the novel methods used and integrated which consist of an enhanced STAKCERT KDD Processes, a STAKCERT worm classification, a STAKCERT worm relational model and a STAKCERT worm apoptosis algorithm. The STAKCERT model has succeeded to fill in all the gaps identified in the existing works, furthermore it has achieved a better accuracy rate compared to the existing works. The new methods prove the effectiveness of the STAKCERT model developed.

In this Chapter the conclusions of the research are discussed by summarising the main contributions that have been made and possible directions for future work that could be undertaken as a way forward with regards to continuing the research in this area.

6.1 Main Contributions

I) STAKCERT Worm Classification and Relational Model.

A good understanding of the worm architecture is a must, prior to the creation of a worm detection and response model. This is not only limited to the worm's structure, but also by considering the threats it poses, the way it spreads, and the survival methods it uses to avoid being detected by anti-virus software. Therefore a thorough study and experimentation with regard to worm architecture are conducted, and a related correlation is made which leads to the new formation of the STAKCERT worm classification and the STAKCERT worm relational model. The STAKCERT worm relational model is based on the STAKCERT worm classification. Both of them have a similar goal which is to make worm detection easier and more effective. They are a part of the whole STAKCERT model.

The STAKCERT worm classification consists of five main attributes - the payload, infection, activation, propagation and operating algorithm. No matter what kind of worm variations have been introduced, based on these five main attributes, the new or existing worm can be easily categorised into a different type of worm group. Once the worm group identified, the detection and removal steps can easily be applied to the new worm variation. Even as time evolves, it is believed that these five main attributes are vital in deciding what kind of techniques should be applied in terms of the worm detection and removal steps. These five main worm attributes are of great value and can help to reduce the response time and the solution provided is more accurate than previous methods.

II) Enhanced STAKCERT KDD Processes.

A good and efficient model is built using a comprehensive methodology. For this thesis, an enhanced and comprehensive methodology for worm analysis, starting from data pre-processing and moving through to post-processing is developed. This methodology is called the STAKCERT KDD processes which can be seen in Figure 3.5 in Chapter 3. One of the most common problems faced by the virus security analyst or the researcher in the worm research area, is how and where to start analysing the worm. In this STAKCERT KDD processes, the original KDD processes are enhanced by integrating the worm analysis together with standard operating procedures (SOP) in incident response, statistical analysis, security metrics and data mining. As far as the researcher is aware, this is a novel approach to addressing the issues raised. This methodology offers a good point of reference for future work in worm research. This methodology not only helps to reduce analysis time, it also helps to improve the worm detection and worm response accuracy results, an outcome which has been indicated by this thesis.

III) STAKCERT model for worm detection outperforms existing works with a better accuracy.

A comparison with existing worm detection work was conducted to test the efficiency of STAKCERT model for worm detection. The performance criteria for worm detection consist of overall accuracy rate, TP rate (TPR), FP rate (FPR) and FN rate (FNR). Based on the experimentation conducted, the STAKCERT model yielded 98.75% of overall accuracy rate, 98.88% of TPR, 0.2% of FPR and 1.45% of FNR. In comparison, Dai *et al.* (2009), yielded 93.2% overall accuracy rate, 91.9% TPR, 9.6% FPR and 6.8% FNR while Siddiqui *et al.* (2009) yielded 95.6% overall accuracy rate, 96% TPR, 3.8% FPR and 4.4% FNR. The STAKCERT model results have outperformed these two approaches in terms of higher overall accuracy and TPR and lower FPR and FNR. This thesis has accomplished one of its objectives which is to improve the existing worm detection technique with a better accuracy. In fact, this thesis overall accuracy rate is 3.15% higher than Siddiques *et al.* (2009). These encouraging results produced by the STAKCERT model have been achieved with the help of the STAKCERT worm classification and STAKCERT relational model and by applying the STAKCERT KDD process as part of the methodology.

IV) Apoptosis as a new technique for worm response.

From an incident response perspective, comparisons were made with the existing works of Kim *et al.* (2010) and Liu *et al.* (2010), where both had the same objective, which was to detect and respond to an incident. The improvement made in the STAKCERT model compared to these two works was to add one further new step. This was applying apoptosis during the response process and altering the scope of implementation, whereby the STAKCERT model was dedicated specifically to detecting and responding to a worm. Apoptosis, also known as cell-programmed death, is a concept borrowed from the human immunology system, where once a cell has been identified as being severely infected by a virus, it destroys itself. In the worm response context, apoptosis is implemented by disconnecting the infected computer from the network or from the Internet to avoid the worm propagating to other computers.

V) STAKCERT worm apoptosis algorithm.

Prior to the formation of the STAKCERT worm apoptosis algorithm, analysis and experimentation are conducted to identify the most important features for triggering apoptosis. As a result, weight and severity have been identified as the most important features. Consequently, an algorithm called the STAKCERT worm apoptosis algorithm has been formed. The weight, severity and STAKCERT worm classification are part of this algorithm. The five main attributes extracted from the STAKCERT worm classification are the payload, infection, activation, propagation and operating algorithm and these were later assigned with weight and severity values. Moreover, the security metrics have been used as a method for quantifying and assigning the weight and severity values. Based on the experimental results, the assigned rate of severity was 100% accurate.

VI) *STAKCERT model for worm response with an accuracy rate.*

This thesis cannot directly compare the accuracy result with existing work for worm response, since none of the existing work provides any accuracy results. Still it is possible to make a comparison with the existing works by comparing the methods they applied in their works. Based on the comparison made under section 5.5, applying the standard operating procedures in incident response, security metrics and apoptosis to STAKCERT model, help to give a good performance for worm response. To show these methods are working effectively, the STAKCERT model was simulated with the eradication solutions and yielded an overall accuracy rate of 98.08% and F-measure rate of 100%. These results indicate that the STAKCERT model is an effective worm response model. With these results, this thesis final objective which is to provide an accuracy rate for worm response has been accomplished, which can be used as a reference in future.

6.2 Future Work

For future work, broaden plans have been made to the scope of this research which currently focuses on worms based on Windows operating system. This provides greater opportunities to explore the different types of malicious code and use of STAKCERT model on different platforms. Furthermore, an expansion of the different types of malicious code would improve the robustness of the STAKCERT model (although a few modifications would have to be implemented under the pre-processing procedures). In addition, the integration of dynamic and static analyses needs more investigation and refinement to produce improved results for worm detection.

In Chapter 5, the STAKCERT model was simulated with the plan that it can be applied in worm detection and response software in the future. There are a few challenges that can be met to make this model work more effectively in software implementation. Firstly, the integration of the intelligent system concept, with the aim of providing a better eradication solution and to reduce false alarms. The retrieval method could be improved to ensure the accuracy and solution return is 100%. Secondly, future software must be secure from any intrusion by integrating the software integrity check. Lastly it will have to have a centralised repository to save all the worm detection and response descriptions and accuracy results, which later can be accessed all over the world. Then other researchers can measure the effectiveness of their works by referring to this repository and make comparison with this thesis results.

These suggested improvements could greatly enhance current solutions aimed at handling worm detection and response.

In addition, since one of the thesis future works is to develop software, an effective approach to teach end users how to use this software is desirable.

Therefore, another future work planned is to apply the worm detection and response techniques software to video games. This is to help the end user visualizes how the complete process is carried out; from pre-processing until the post-processing for worm detection, subsequently together with the worm response. Furthermore, to make the video game more interactive, the end user will be given the opportunity to get involved in performing the worm analysis and consequent implications will be seen if the procedures while performing the task were not being done properly. The more interesting part is the visualisation of the apoptosis into worm response since apoptosis is part of human immunology study. To make this video game a success, thorough studies need to be carried out in the first place. These include applying the appropriate methodology, graphics and integrating data mining to make the video game more interactive and intelligent and as well as optimising the game performance. Even though not much research in applying security into video games has been conducted for the past few years, still there are few works carried out in such area for example by Cone *et al.* (2007), who built CyberCIEGE to support education and training in computer and network security. Indeed a part of the video game consists of an overview of malicious code, which caught the researcher attention and the researcher plans to broaden this scope for future work. This work can be used as guidance and basis in developing this interactive video game. It is believed that applying video game within security context has a bright potential and is a promising field to be explored more in the future.

REFERENCES

A. Somayaji, S. Hofmeyr and S. Forrest. (1998). Principles of a computer immune system. *In Meeting on New Security Paradigms*, 23-26 Sept. 1997, Langdale, UK, USA : ACM ,pages 75–82.

Adleman,L. (1990). An Abstract Theory of Computer Viruses. Advances in Cryptology — CRYPTO' 88, Springer Berlin / Heidelberg. 403, pp.354-374.

Agosta,J.M., Diuk-Wasser,C., Chandrashekar,J. and Lividas,C. (2007). An adaptive anomaly detector for worm detection. *In Proceedings of the 2nd USENIX workshop on Tackling computer systems problems with machine learning techniques (SYSML'07),* pp. 1-6.

Albanese, D.J., Wiacek,M.J., Salter,C.M. and Six,J.A. (2004). The Case for Using Layered Defenses to Stop Worms (Report style)," UNCLASSIFIED-NSA Report, pp.10-22.

Anton, D. (2009). *Win32.Worm.Viking.BU* [online]. Available from: http://www.bitdefender.com/VIRUS-1000337-world--Win32.Worm.Viking.BU.html [Accessed 26th June 2011].

Atzeni,A. and Lioy,A. (2006). Why to adopt a security metric? A brief survey. In: Gollmann,D., Massacci,F. and Yautsiukhin,A. (eds.), *Quality of Protection Security Measurements and Metrics*, USA: Springer, pp1-12.

Bailey, M., Cooke, E., Jahanian, F., Watson, D. and Nazario, J. (2005). The Blaster Worm: Then and Now, *IEEE Security & Privacy*, vol.3, no.4, pp. 26-31.

Bejtlich, R., Steven, J. and Peterson,G. (2011). Directions in Incident Detection and Response. *IEEE Security and Privacy*, vol. 9, no. 1, pp. 91-92

Berghel,H. (2001). The Code Red Worm: Malicious software knows no bounds, *Communication of the ACM*, vol.44, no.12, pp.15-19.

B. D. Cone, C. E. Irvine, M. F. Thompson and T. D. Nguyen. (2007). A video game for cyber security training and awareness, *Computers & Security*, vol. 26, pp. 63-72.

Ben Othmane, L. and Lilien, L. (2009). Protecting Privacy of Sensitive Data Dissemination Using Active Bundles. *Privacy, Security, Trust and the Management of e-Business*, pp. 202-213.

Bradley, T. (2010). *Microsoft Reveals Stuxnet Worm Exploits Multiple Zero Days* [online], Available from: http://www.pcworld.com/businesscenter/article/205465/microsoft_reveals_stuxnet_worm_exploits_multiple_zero_days.html [Accessed 26th June 2011]

BSI. (1999). Information security management, BS7799, part 1: code of practice for information security management'.

Castaneda, F. , Can Sezer,E.and Xu,J. (2004) WORM vs. WORM: preliminary study of an active counter-attack mechanism, *Proceedings of the 2004 ACM workshop on Rapid malcode,* October 29-29, 2004, Washington DC.

CERT. (2002). CERT Advisory CA-2002-25 Integer Overflow in XDR Library, Available: http://www.cert.org/advisories/ca-2002-25.html.

Christoffersen,D. and Mauland, B.J. (2006). *Worm Detection Using Honeypots* Master dissertation, Norwegian University of Science and Technology.

Cheetancheri, S.G. (1998). Modeling a computer worm defense system, Master dissertation, University of California.

Cisco Systems,Inc. (2004). *Code-Red Worm Origins and Evolution* [online]. Available from:
http://www.cisco.com/warp/public/cc/so/cuso/epso/sqfr/scdam_wp.htm
[Accessed: 26th June 2011]

Cohen, F.B. (1985). *Computer viruses*. PhD thesis, University of Southern California.

Cohen, F.B. (1987). Computer viruses : Theory and experiments. Computers & Security 6(1), pp.22-35.

Cohen, F. B. (1992). A formal definition of computer worms and some related results. *Computer. Security,* **11** (7), pp. 641-652.

CyberSecurity Malaysia. (2010). *MyCERT Incident Statistics* [online]. Available from:
http://www.mycert.org.my/en/services/statistic/mycert/2010/main/detail/725/index.html [Accessed: 26th June 2011]

Dabirsiaghi, A. (2008). Building and Stopping Next Generation XSS Worms. *Proceedings of the 3rd International OWASP Symposium on Web Application Security Conference Europe 2008.*

Dagon,D., Qin,X.,Gu,G., Lee,W., Grizzard,J.,Levine,J. and Owen,H. (2004). HoneyStat: Local Worm Detection Using Honeypots. *Recent Advances in Intrusion Detection*. Vol. 3224/2004, Springer Berlin / Heidelberg, pp. 39-58.

Dai,J., Guha,R. and Lee,J. (2009). Efficient Virus Detection Using Dynamic Instruction Sequences. *Journal of Computers.* Vol 4, No 5, pp. 405-414.

Dasgupta, D. (1997). Artificial Neural Networks and Artificial Immune Systems: Similarities and Differences. *Proc. of the IEEE SMC.*, 1, pp. 873-878.

Dasgupta, D. (1999). Immunity-Based Intrusion Detection System: A General Framework. In *Proc. of the 22nd NISSC.*

Ellis, D. (2003). Worm anatomy and model. *Proceedings of the 2003 ACM workshop on Rapid malcode*, Washington, DC, USA, ACM pp. 42-50. Available from: http://portal.acm.org/citation.cfm?id=948196 [Accessed: 7th May 2011]

Eset Company. (2010). *Globat Threats Trend* [online]. Available from: http://www.eset.ie/threat-center/case_study/GlobalThreatTrendsJanuary2010.pdf [Accessed: 26th June 2011]

Fayyad, U., Piatetsky-Shapiro, G. and Smyth, P. (1996). The KDD Process for Extracting Useful Knowledge Data. *Communications of the ACM*, v. 39(no. 11): pp. 27-34.

Filiol, E.,Helenius,M. and Zaner,S. (2006). Open Problems in Computer Virology. *Journal in Computer Virology,* 1 (3),Springer Paris, pp. 55-66. Available from: 10.1007/s11416-005-0008-3 [Accessed: 26th June 2011].

Fisch, D., Hofmann, A. and Sick,B. (2010). On the versatility of radial basis function neural networks: A case study in the field of intrusion detection. *Information Sciences* 180(12), pp. 2421-2439.

Fitzgerald,N. (2008). *What is a Worm?* (Computer virus) [online]., Available from:
http://stason.org/TULARC/security/computer-virus-l/11-What-is-a-Worm-Computer-virus.html [Accessed: 26th June 2011]

Garfinkel,T. and Rosenblum, M. (2003). A Virtual Machine Introspection Based Architecture for Intrusion Detection,*In Proc. Network and Distributed Systems Security Symposium*, pp. 191-206.

Giudici,P. (2010). Data Mining Model Comparison. *Data mining and knowledge discovery*. 2nd edn. New York:Springer, pp.641-654.

Goel, S. and Gangolly, J. S. (2007). On decision support for distributed systems protection: A perspective based on the human immune response system and epidemiology. *International Journal of Information Management,* 27 (4), pp. 266-278.

Greasley, P. (2008). *Quantitative data analysis using SPSS: An introduction for health and social sciences*. Open University Press, McGraw-Hill Education, Glasgow, pp. 63.

Grzymala-Busse, J.W. (2010). Rule induction. In: O.Maimon, L.Rokach (eds.), *Data Mining and Knowledge Discovery Handbook,* 2nd ed., New York :Springer, pp 249-265.

Hall,M., Frank,E., Holmes,G., Pfahringer,B., Reutemann,P. and Witten,I.H. (2009). The WEKA Data Mining Software: An Update; *SIGKDD Explorations*, Volume 11, Issue 1.

Hawkins,S., C. Yen,D. and C. Chou,D. (2000). Awareness and challenges of Internet security, *Information Management & Computer Security*, Vol. 8 Iss: 3, pp.131 – 143.

Helenius, M. (2002). *A System to Support the analysis of Antivirus Products Virus Detection Capabilities*. PhD Thesis. Department of Computer & Information Sciences, University of Tampere. Available from:http://acta.uta.fi/pdf/951-44-5394-8.pdf [Accessed:7th May 2011]

Henchiri, O. and Japkowicz, N. (2006). A Feature Selection and Evaluation Scheme for Computer Virus Detection. *Proceedings of the Sixth International Conference on Data Mining, 2006. ICDM '06.* Hong Kong: IEEE Xplore, pp. 891 - 895. Available from: http://doi.ieeecomputersociety.org/10.1109/ICDM.2006.4 [Accessed: 26th June 2011]

Hively,L., Sheldon,F. and Squicciarini,A. (2010). A Vision for Scalable Trustworthy Computing, *IEEE Security and Privacy*, IEEE computer Society Digital Library. IEEE Computer Society, Available from: http://doi.ieeecomputersociety.org/10.1109/MSP.2010.142 [Accessed:7th May 2011]

Hofmeyr S. A. and Forrest, S. (1999). Immunity by Design: An Artificial Immune System. *Proc. Of GECCO'99*, pp. 1289-1296.

Hunt, J. E. and Cooke, D. E. (1996). Learning Using an Artificial Immune System. *Journal of Network and Computer Applications*, 19, pp. 189-212.
Kim,H.K., Im,K.H, and Park, S.C. (2010). DSS for computer security incident response applying CBR and collaborative response. *Expert Systems with Applications*, 37, pp. 852–870

Ishizaki, Y., L. Cheng, A.W. Mudge and M.C. Raff. (1995). Programmed Cell Death by Default in Embryonic Cells, Fibroblasts and Cancer Cells. *Mol. Bio. Cell*, 6(11). pp.:1443-1458

Jaquith, A. (2007). *Security metrics: replacing fear, uncertainty and doubt*. United States of America: Addison-Wesley. p40.

Jordan, C. (2005). Writing detection signatures, *USENIX ;login:*, vol. 30, no. 6, pp. 55–61.

Kaspersky. (2008). *Network worm* [online]. Available from: http://www.viruslist.com/en/virusesdescribed?chapter=152540408 [Accessed:26th June 2011]

Kaspersky (2011). *Kaspersky Lab Publishes Threat Evolution Report for 2010* [online]. Available from: http://www.securelist.com/en/analysis/204792161/Kaspersky_Security_Bulletin_Malware_Evolution_2010 [Accessed 26th June 2011]

Kerr, J.F.R, Winterford, C.M. and Harmon, B.V. (1994). Apoptosis: Its Significance in Cancer and Cancer Therapy. *Cancer*, 73 (8), pp. 2013-2026

Khan,H., Mirza,F. and Khayam,S.A. (2010). Determining malicious executable distinguishing attributes and low-complexity detection. *Journal In Computer Virology*. 7(2), pp. 95-105

Kienzle,D.M. and Elder, M.C. (2003). Recent Worms: A Survey and Trends, *Proceedings of the 2003 ACM workshop on Rapid malcode (WORM '03),* New York,USA. pp. 1-10.

Killcrece,G., Kossakowski, K.P, Ruefle,R. and Zajicek,M. (2003). Organizational Models for Computer Security Incident Response Teams (CSIRTs), Carnegie Mellon University: CMU/SEI-2003-HB-001

Kim,H.K., Im,K.H and Park, S.C. (2010). DSS for computer security incident response applying CBR and collaborative response. *Expert Systems with Applications*, 37, pp. 852–870.

Kohavi, R. and Provost F. (1998). Glossary of terms. *Machine Learning Journal*, 30, pp. 271-274.

Kovalerchuk, B. and Vityaev,E. (2010). Data Mining for Financial Applications. In: Maimon, Oded; Rokach, Lior ,eds. *Data mining and knowledge discovery*. 2nd edn. New York:Springer, pp. 1154-1169.

Kruegel, C., Kirda, E., Mutz D., Robertson W. and Vigna, G. (2005). Polymorphic Worm detection using structural information of executables, *8th International Symposium on Recent Advances in Intrusion Detection (RAID).*

Laborger, P. (19th October 2005). *XSS worm hits Myspace* [online]. Available from:http://www.securityfocus.com/brief/18 [Accessed:26th June 2011]

Lavrac, N. and Zupan,B. (2010). Data Mining in Medicine, In: Maimon, Oded; Rokach, Lior ,eds. *Data mining and knowledge discovery*. 2nd edn. New York:Springer, pp. 1111-1136.

Levin, J., Labella,R.,. Owen, H., Contis, D. and Culver, B. (2003). The Use of Honeypots to Detect Exploited Systems Access Across Large Enterprise Networks. *Proceedings of the 2003 IEEE Workshop on Information Assurance*, USA, 18-20th June 2003, IEEE Xplore, pp.92-99.

Liu, P., H. Yu and Ma, G. (2010). An incident response decision support system based on CBR and ontology. *International Conference on Computer Application and System Modeling (ICCASM) Proceedings,* 22nd-24th Oct 2010. IEEE Xplore. pp. 337-340.

Liu,Z. and Uppala,R. (2006). A Dynamic Countermeasure Method for Large-Scale Network Attacks, *2nd IEEE International Symposium on Dependable, Autonomic and Secure Computing*, pp.163 - 170.

Maimon, O. and Rokach, L. (2010) Introduction to Knowledge Discovery and Data Mining In: Maimon, Oded; Rokach, Lior ,eds. *Data mining and knowledge discovery*. 2nd edn. New York:Springer,pp 1-15.

Maynor, D., Mookhey,K.K.,Cervini,J. and Roslan,F. (2007). Metasploit Toolkit for Penetration Testing, Exploit Development, and Vulnerability Research, USA: Syngress Publishing,Inc.

Microsoft. (2008). *4 steps to protect your computer* [online]. Available from: http://www.microsoft.com/protect/computer/default.mspx [Accessed:26th June 2011]

Microsoft Corporation (2011). *Worm:Win32/Stuxnet.A* [online]. Available from: http://www.microsoft.com/security/portal/Threat/Encyclopedia/Entry.aspx?Name =Worm%3aWin32%2fStuxnet.A [Accessed 26th June 2011]

Middlemiss, M. J. and Dick, G. (2003). Weighted feature extraction using a genetic algorithm for intrusion detection. In *Proceedings of the evolutionary computation,* Vol. 3, pp. 1669–1675.

Miles, S. G (2001). *Incident Response Part #2: Identification* [online]. Available from:
http://www.securityhorizon.com/whitepapersTechnical/IncidentResponsepart2.p df [Accessed: [Accessed:26th June 2011]

Mitropoulos, S., Patsos, D. and Douligeris, C. (2006). On Incident Handling and Response: A state-of-the-art approach. *Computers & Security,* 25 (5), pp.351-370.

Moskovitch, R., Y. Elovici and Rokach,L. (2008a). Detection of unknown computer worms based on behavioral classification of the host*. Computational Statistics & Data Analysis* 52(9). pp.4544-4566.

Moskovitch,R., Stopel,D., Feher,C., Nissim,N., Japkowicz, N. and Elovici,Y. (2008b). Unknown malcode detection and the imbalance problem. *Journal In Computer Virology.* Volume 5, Number 4, 295-308, DOI: 10.1007/s11416-009-0122-8.

Mulholland, C. L., Sterritt,R., O'Hagan,P. and Hanna,E. (2008). Tagging and Tracking System for Prisons and Correctional Facilities- A Design Roadmap. *Fifth IEEE Workshop on Engineering of Autonomic and Autonomous Systems*, *EASE 2008*, pp. 143-153.

MyCERT (2002). Computer *Worm* Incident Handling *Standard Operating Procedure* [online]. Available from: http://www.mycert.org.my/en/services/advisories/mycert/2002/main/detail/111/in dex.html [Accessed: [Accessed:26th June 2011]

MYCERT. (2009a). *MA-151.032009: MyCERT Special Alert - Conficker.C Worm* [online]. Available from: http://www.mycert.org.my/en/services/advisories/mycert/2009/main/detail/647/in dex.html [Accessed:26th June 2011]

MYCERT. (2009b). *Tips to Safeguard Yourself from Fraudulent Email / Phishing Attempts* [online]. Available from: http://www.mycert.org.my/en/resources/email/email_tips/main/detail/513/index.html [Accessed:26th June 2011]

Nachenberg,C. (2000). The Evolving Virus Threat, *23rd NISSC Proceedings*," Baltimore, Maryland.

Nazario, J., Anderson,J., Wash,R. and Conelly,C. (2001). The Future of Internet Worms, *In Proceedings Black Hat Conference,* USA. Available from: http://www.cgisecurity.com/lib/the_future_of_internet_worms.pdf.[Accessed: 26th June 2011].

Nicol, D. M. (2005). Modeling and simulation in security evaluation.*Security & Privacy*, IEEE 3(5), pp.71-74.

Olsen, M.M., Danieli, N. S. and Siegelmann,H.T. (2008). Robust artificial life via artificial programmed death. *Artificial Intelligence,* 172 (6-7), pp. 884-898.

Piatetsky-Shapiro, G. (1991). Knowledge Discovery in Real Databases: A Report on the IJCAI-89 Workshop. *AI Magazine* 11(5): 68–70.

Pricewaterhouse Coopers,LLP. (2010). *2010 Information Security Breaches Survey* [online]. Technical Report UK Department of Business, Enterprise and Regulatory Reform (BERR), Available from: http://www.infosec.co.uk/files/isbs_2010_technical_report_single_pages.pdf [Accessed:26th June 2011].

Prosise, C., Mandia,K. and Pepe,M. (2003).*Incident Response and Computer Forensics*, Second Edition, McGraw-Hill, p15.

Purchon, N. (2000). *Animal Cells* [online]. Available from:(http://www.purchon.com/biology/animal.htm) [Accessed:26th June 2011]

Raff, Martin. (1998). Cell Suicide for Beginners.[online] *Nature* 396(6707). p.119. Available from: http://www.nature.com/nature/journal/v396/n6707/full/396119a0.html [Accessed:26th June 2011]

Quinn, K J Spike. (2010). 2010 New Zealand Computer Crime and Security Survey [online]. Available from: http://hdl.handle.net/10523/1455 [Accessed:26th June 2011]

Reese, R.L.R. (2003). Incident Handling: An Orderly Response to Unexpected Events. *Proceedings of the 31st annual ACM SIGUCCS conference on User services (SIGUCCS 03).* Texas, USA ,21-24 Sept 2003, USA :ACM New York, pp. 97-102.

Riordan, J. and Alessandri, D. (2000). Target Naming and Service Apoptosis. *Recent Advances in Intrusion Detection*, Springer Berlin / Heidelberg. 1907.pp.217-225.

Robert, A. (4th December 2009). *Twitter response to XSS worm attack* [online]. Available from: http://www.cgisecurity.com/2009/04/twitter -response-to-xss-worm -attack.html [Accessed:26th June 2011]

Sadasivam, K., Samudrala, B. and Yang,T.A. (2005). Design of network security projects using honeypots. *Journal of Computing in Small Colleges.*, 20(4), pp. 282-293.

SANS Institute. (2008). Security 504.1 Incident Handling Step-by-Step and Computer Crime Investigation. SANS Institute.

Saudi, M.M. (2005). Combating Worms Outbreaks: Malaysia Experience. *International Journal of Learning.* Volume 12, Issue 2, pp.295-304.

Saudi,M.and Jomhari,N.(2006). Knowledge Structure on Virus for User Education. *2006 International Conference on Computational Intelligence and Security*, pp. 1515 - 1518.

Saudi, M. M., Woodward, M., Cullen, A. J. and M. Noor, H. (2008a). An overview of apoptosis for computer security. *International Symposium on Information Technoloogy 2008(ITSim2008), Conference Proceedings,* ,v 4, Kuala Lumpur, Malaysia, 26-28th August 2008, IEEE XPlore, pp. 2534-2539.

Saudi, M. M, M.Isa, M.A., Cullen, A.J., Woodward, M. and M. Noor, H. (2008b). Defending Virus Infection through DVItApoptosis System, *4th International Conference on Information Technology and Multimedia,Conference Proceedings*, Kuala Lumpur,Malaysia. 18-19th Nov 2008.

Saudi,M. M., M.Tamil, E., Cullen,A.J., Woodward, M., I.Idris,M.Y.(April 2009) Reverse Engineering: EDOWA Worm Analysis and ClassificationAdvances in *Electrical Engineering and Computational Science, Lecture Notes in Electrical Engineering.* . In: Ao,S.I.& Gelman,L.,eds. Berlin: Springer Netherlands, pp. 277-288.

Saudi, M.M, Cullen, A.J. and Woodward, M.E. (2010a). STAKCERT Worm Relational Model for Worm Detection, *Lecture Notes in Engineering and Computer Science:* Proceedings of The World Congress on Engineering 2010, WCE 2010, 30 June - 2 July, 2010, London, U.K. pp 469-473.

Saudi, M.M, Cullen, A.J. and Woodward, M.E. (2010b). Statistical Analysis in Evaluating STAKCERT Infection, Activation and Payload Methods, *Lecture Notes in Engineering and Computer Science*: Proceedings of The World Congress on Engineering 2010, WCE 2010, 30 June - 2 July, 2010, London, U.K. pp 474-479.

Scharenbroich, L. (27th August 2003). *ROC Curves* [online]. Available from: http://woldlab.caltech.edu/docs/MLX/user/node68.html [Accessed:26th June 2011]

Schneier,B.(2005). Attack Trends: 2004 and 2005, *Queue*, v.3 n.5.

Schultz, E.E. and Shumway, Russell. (2001). *Incident Response: A Strategic Guide to Handling System and Network Security Breaches*, 1st edn., United States of America: New Riders Publishing.

Schultz, M. G., Eskin,E., Zadok,E. and Stolfo,S.J. (2001). Data Mining Methods for Detection of New Malicious Executables. *In Proceedings of the 2001 IEEE Symposium on Security and Privacy*, IEEE Computer Society, pp 38, [Accessed:26th June 2011]

Schultz, E. E. (2007). Computer forensics challenges in responding to incidents in real-life settings. *Computer Fraud & Security,* 2007(12), pp. 12-16.

Schroeder, C. (2005). *Home Networks, Safe or Sane?* [online]. CA: Canaudit,Inc. Available from: http://www.canaudit.com/Perspectives/Volume6_Issue3.pdf [Accessed:26th June 2011]

Schwetzer, D. (2003). *Incident Response: Computer Forensics Toolkit*, Indianapolis: Wiley Publishing, Inc.

Shannon, C. and Moore,D. (2004). The Spread of the Witty Worm. *IEEE Security & Privacy*, vol.2, no.4, pp.36-50.

Shearer,J. (2010). *W32.Stuxnet* [online]. Available from: http://www.symantec.com/security_response/writeup.jsp?docid=2010-071400-3123-99 [Accessed:26th June 2011]

Shepherd, S. A. (22nd April 2003). Vulnerability disclosures [online]. Source:Sans Institute Infosec reading room, Available from:http://www.sans.org/reading_room/whitepapers/threats/how_do_we_define_responsible_disclosure_932?show=932.php&cat=threats
[Accessed:26th June 2011]

Siddiqui, M., Wang,M.C. and Lee,J. (2009). Detecting Internet Worms Using Data Mining Techniques, *Journal of Systemics, Cybernetics and Informatics*, vol. 6, no. 6, pp. 48–53.

Siddique, F. and Maqbool, O., (2011). Analyzing Term Weighting Schemes for Labeling Software Clusters, *15th European Conference on Software Maintenance and Reengineering Proceedings*, pp.85-88.

Singhal, A. and Jajodia,S. (2010). Data Mining for Intrusion Detection, In: Maimon, Oded; Rokach, Lior ,eds. Data mining and knowledge discovery. 2nd edn. New York:Springer, pp.1171-1180.

Skoudis, E. and Zelster,L. (2004). *Malware Fighting Malicious Code*, Pearson Education,Inc., New Jersey, pp.71-88.

Smith, B. (2008). A Storm (Worm) Is Brewing, *Computer,* Vol.41 Iss:2, pp.20 – 22.

Spitzner, L. (2003). Honeypots: catching the insider threat, *Proc. 19th Annual Computer Security Applications Conference*, IEEE Xplore, pp. 170-179.

Stopel, D., Moskovitch,R., Boger,Z., Shahar,Y. and Elovici,Y. (2009). Using artificial neural networks to detect unknown computer worms. *Neural Computing &amp; Applications* 18(7), pp. 663-674.

Sterritt, R. (2011). Apoptotic Computing: Programmed Death by Default for Computer-Based Systems. *Computer*. 44(1). pp. 59-65.

Swets, J. (1988). Measuring the accuracy of diagnostic systems. *Science*, 240(4857), pp.1285-1293.

Su, M.-Y. (2011). Real-time anomaly detection systems for Denial-of-Service attacks by weighted k-nearest-neighbor classifiers. *Expert Systems with Applications* 38(4), pp. 3492-3498.

Sullivan, J. A.. (1994). CELLS alive! [online]. Available from:http://www.cellsalive.com/cells/3dcell.htm) [Accessed:26th June 2011]

Symantec Corporation. (2003). *W32.Welchia.Worm* [online]. Available from: http://www.symantec.com/security_response/writeup.jsp?docid=2003-081815-2308-99 [Accessed:26th June 2011]

Tarakanov, A. O. (2008). Immunocomputing for intelligent intrusion detection. *Computational Intelligence Magazine.* IEEE. 3(2). pp. 22-30.

Tewolde, A. (2011). *Analysing the output* [online]. Available from:http://wekadocs.com/node/13 [Accessed:26th June 2011]

Thearling, K. (2010). Data Mining for CRM, In: Maimon, Oded; Rokach, Lior ,eds. Data mining and knowledge discovery. 2nd edn. New York:Springer, pp.1181-1188.

Trend Micro, Inc. (2008). *Top 20 Viruses in history* [online]. Available from: http://us.trendmicro.com/imperia/md/content/us/pdf/aboutus/20thanniversary/top20_viruses.pdf [Accessed:26th June 2011]

Tschudin, C. (1999). Apoptosis — the Programmed Death of Distributed Services. *Secure Internet Programming,* Springer Berlin/Heidelberg, pp.253-260, Available from: http://www.springerlink.com/content/a521221330lvj426/ [[Accessed:26th June 2011]

Tseng, S.-S. and S.-C. Lin. (2009). VODKA: Variant objects discovering knowledge acquisition. *Expert Systems with Applications***.** 36(2, Part 1). pp. 2433-2450.

Tsow, A. (2006). Phishing With Consumer Electronics: Malicious Home Routers,*15th International World Wide Web Conference Proceedings (WWW2006)*, Edinburgh, Scotland.

University of Florida (UF). (2010). *UF IT Security Incident Response Procedures, Standard and Guidelines* [online]. Available from:http://www.it.ufl.edu/policies/security/uf-it-sec-incident-response.html#class [Accessed:26th June 2011]

Vamosi, R. (4th December 2008). *Koobface virus hits Facebook* [online]. Source: CNET news: Security. Available from:http://news.cnet.com/koobface-virus-hits-facebook/ [Accessed:26th June 2011]

Vasudevan, A. (2008). MalTRAK: Tracking and Eliminating Unknown Malware, *Proceedings of the 2008 Annual Computer Security Applications Conference*, Anaheim, CA, 8-12 December 2008, IEEE Xplore, pp 311-321, Available from: 10.1109/ACSAC.2008.44 [Accessed:26th June 2011]

VX Heavens website. (2009). Virus Collection [online]. Available from: http://vx.netlux.org/vl.php. [Accessed:26th June 2011]

Wang, Y.-M., Beck, D., Jiang, X. and Roussev, R. (2005). Automated Web Patrol with Strider HoneyMonkeys: Finding Web Sites that Exploit Browser Vulnerabilities. Microsoft Research, Technical Report MSR-TR-2005-72.

Weaver, N., Paxson, V., Staniford,S. and Cunningham, R. (2003). A Taxonomy of Computer Worms, *Proceedings of the ACM CCS Workshop on Rapid Malcode (WORM)*, pp.11-18.

Weisstein, E. W. (2011). Fisher's Exact Test [online].From MathWorld--A Wolfram Web Resource. Available from: http://mathworld.wolfram.com/FishersExactTest.html [Accessed:26th June 2011]

Werlinger,R., Muldner,K., Hawkey,K.and Beznosov,K. (2010) Preparation, detection, and analysis: the diagnostic work of IT security incident response, *Information Management & Computer Security*, Vol. 18 Iss: 1, pp.26 - 42

White, S. R. (1998). Open problems in computer virus research. *Proceedings of the Virus Bulletin Conference*, Munich Germany, 22 Oct 1998. Available from: http://www.research.ibm.com/antivirus/SciPapers/White/Problems/Problems.html [Accessed:26th June 2011]

White, G. and Granado, N (2009) Developing a Community Cyber Security Incident Response Capability, System Sciences, 2009. *Proceedings 42nd Hawaii International Conference on HICSS '09*, pp. 1-9

Whitty, B. (2007) *Why do People Create Computer Viruses?* [online].Technibble.com
Available from:http://www.technibble.com/why-do-people-create-computer-viruses/ [Accessed:26th June 2011]

Witten, I.H. and Frank, E. (2005). Data Mining. Practical Machine Learning Tools and Techniques. 2nd ed. San Francisco:Morgan Kaufmann.pp.107-108.

Zou, C.C., Towsley, D. and Gong, W. (2004). Email worm modeling and defense, *Computer Communications and Networks, ICCCN 2004*, pp.409-414.

APPENDICES

APPENDIX A: CHI-SQUARE TESTS AND SYMMETRIC MEASURE RESULTS

Chi-square and symmetric measure results

Null hypothesis(H_o) and alternative hypothesis(H_a)	Chi-square tests		Symmetric measure	Conclusion
	Pearson Chi-square value	p value	Phi value	
Finding 1: Relationship between Vulnerability and Email. H_0 = There is no relationship between vulnerability and email. H_a = There is relationship between vulnerability and email.	39.961	0.00	0.498	As a result, the H_0 is rejected and H_a is accepted since the p value is less than 0.05. This indicates that the relationship did not happen by chance, which is based on the Chi-Square tests. The value of the probability (p) for the distribution occurs by chance is 0.00 (refer to Table 1). As a conclusion, there is a positive strong relationship between vulnerability and email.
Finding 2. Relationship between Vulnerability and File. H_0 = There is no relationship between vulnerability and file. H_a = There is relationship between vulnerability and file.	7.835	0.005	-0.221	Therefore, the H_0 is rejected and H_a is accepted since the p value is less than 0.05. This indicates that the relationship did not happen by chance, which is based on the Chi-Square tests. The value of the probability (p) for the distribution occurs by chance is 0.005. The result of the analysis is summarised in Table 2. As a conclusion, there is a negative weak relationship between vulnerability and file.

Finding 3. Relationship between Email,Vulnerability and File. H_0 = There is no relationship between vulnerability, file and email. H_a = There is relationship between vulnerability, file and email.	16.460	0.128	0.227	The relationship that would like to be tested is email influencing the vulnerability and the file. Based on the statistical analysis conducted, email did not influence the vulnerability and file. In the table Chi Square and symmetric measure, the 'Yes' column is being referred. The Pearson Chi-Square value is 16.460, significance or probability (p) value of 0.128 and Phi value is 0.227 using the Chi-Square tests and symmetric measure. Based on the result analysis that is summarised in Table 3, H_0 is accepted since the p value is more than 0.05. Therefore, the relationship might happened by chance with 22.7%. This is calculated by using the Chi-Square equation.
Finding 4. Relationship between Vulnerability and Sharing Directories. H_0 = There is no relationship between vulnerability and sharing directories. H_a = There is relationship between vulnerability and sharing directories.	16.460	0.000	-0.321	Based on the statistical analysis conducted, the relationship between vulnerability and sharing directories has a negative weak relationship with Pearson Chi-Square value is 16.460, significance or probability (p) value of 0.000 and Phi value is -0.321 using the Chi-Square tests and symmetric measure. Therefore, the H_0 is rejected and H_a is accepted since the p value is less than 0.05. Based on the Chi-Square tests, the relationship did not happen by chance. The value of the probability (p) for the distribution occurs by chance is 0.00. The result of the analysis is summarised in Table 3.1. It is concluded that there is a relationship between vulnerability and sharing directories.

Finding 5. Relationship between Self Activation and Human Trigger. H_0 = There is no relationship between self activation and human trigger. H_a = There is relationship between self activation and human trigger.	28.308	0.000	-0.419	Based on the statistical analysis conducted, the relationship between self activation and human trigger has almost a strong negative relationship with Pearson Chi-Square value is 28.308, significance or probability (p) value of 0.000 and Phi value is -0.419 using the Chi-Square tests and symmetric measure. The result of this analysis is summarised in Table 3.2. Since the p value is less than 0.05, the H_0 is rejected and H_a is accepted. This indicates that the relationship did not happen by chance, which is based on the Chi-Square tests. The value of the probability (p) for the distribution occurs by chance is 0.00. As a conclusion, there is a relationship between self activation and human trigger.
Finding 6. Relationship between Autorun Registry and Backdoor. H_0 = There is no relationship between autorun registry and backdoor. H_a = There is relationship between autorun registry and backdoor.	6.630	0.010	0.203	Based on the statistical analysis conducted, the relationship between autorun registry and backdoor has almost positive weak relationship with Pearson Chi-Square with a value of 6.630, significance or probability (p) value of 0.010 and Phi value is 0.203 using the Chi-Square tests and symmetric measure. As a result, the H_0 is rejected and H_a is accepted since the p value is less than 0.05. This indicates that the relationship did not happen by chance, which is based on the Chi-Square tests. The value of the probability (p) for the distribution occurs by chance is 0.010. The result of the analysis is summarised in Table 3.3. As a conclusion, there is a relationship between autorun registry and backdoor.

I) Finding 1. Results for relationship between vulnerability and email.

Vulnerability Exploit * Email Crosstabulation

			Email		
			No	Yes	Total
Vulnerability Exploit	No	Count	96	14	110
		Expected Count	79.3	30.7	110.0
		% within Vulnerability Exploit	87.3%	12.7%	100.0%
		% within Email	82.8%	31.1%	68.3%
	Yes	Count	20	31	51
		Expected Count	36.7	14.3	51.0
		% within Vulnerability Exploit	39.2%	60.8%	100.0%
		% within Email	17.2%	68.9%	31.7%
Total		Count	116	45	161
		Expected Count	116.0	45.0	161.0
		% within Vulnerability Exploit	72.0%	28.0%	100.0%
		% within Email	100.0%	100.0%	100.0%

Chi-Square Tests

	Value	df	Asymp. Sig. (2-sided)	Exact Sig. (2-sided)	Exact Sig. (1-sided)
Pearson Chi-Square	39.961[b]	1	.000		
Continuity Correction [a]	37.610	1	.000		
Likelihood Ratio	38.613	1	.000		
Fisher's Exact Test				.000	.000
Linear-by-Linear Association	39.712	1	.000		
N of Valid Cases	161				

a. Computed only for a 2x2 table

b. 0 cells (.0%) have expected count less than 5. The minimum expected count is 14.25.

Symmetric Measures

		Value	Approx. Sig.
Nominal by Nominal	Phi	.498	.000
	Cramer's V	.498	.000
	Contingency Coefficient	.446	.000
N of Valid Cases		161	

II) Finding 2. Results for relationship between vulnerability and file.

Vulnerability Exploit * File Crosstabulation

			File		
			No	Yes	Total
Vulnerability Exploit	No	Count	43	67	110
		Expected Count	51.2	58.8	110.0
		% within Vulnerability Exploit	39.1%	60.9%	100.0%
		% within File	57.3%	77.9%	68.3%
	Yes	Count	32	19	51
		Expected Count	23.8	27.2	51.0
		% within Vulnerability Exploit	62.7%	37.3%	100.0%
		% within File	42.7%	22.1%	31.7%
Total		Count	75	86	161
		Expected Count	75.0	86.0	161.0
		% within Vulnerability Exploit	46.6%	53.4%	100.0%
		% within File	100.0%	100.0%	100.0%

Chi-Square Tests

	Value	df	Asymp. Sig. (2-sided)	Exact Sig. (2-sided)	Exact Sig. (1-sided)
Pearson Chi-Square	7.835[b]	1	.005		
Continuity Correction [a]	6.913	1	.009		
Likelihood Ratio	7.877	1	.005		
Fisher's Exact Test				.007	.004
Linear-by-Linear Association	7.786	1	.005		
N of Valid Cases	161				

a. Computed only for a 2x2 table

b. 0 cells (.0%) have expected count less than 5. The minimum expected count is 23.76.

Symmetric Measures

		Value	Approx. Sig.
Nominal by Nominal	Phi	-.221	.005
	Cramer's V	.221	.005
	Contingency Coefficient	.215	.005
N of Valid Cases		161	

III) Finding 3. Results for relationship between vulnerability, file and email.

Vulnerability Exploit * File * Email Crosstabulation

Email				File		Total
				No	Yes	
No	Vulnerability Exploit	No	Count	32	64	96
			Expected Count	38.9	57.1	96.0
			% within Vulnerability Exploit	33.3%	66.7%	100.0%
			% within File	68.1%	92.8%	82.8%
		Yes	Count	15	5	20
			Expected Count	8.1	11.9	20.0
			% within Vulnerability Exploit	75.0%	25.0%	100.0%
			% within File	31.9%	7.2%	17.2%
	Total		Count	47	69	116
			Expected Count	47.0	69.0	116.0
			% within Vulnerability Exploit	40.5%	59.5%	100.0%
			% within File	100.0%	100.0%	100.0%
Yes	Vulnerability Exploit	No	Count	11	3	14
			Expected Count	8.7	5.3	14.0
			% within Vulnerability Exploit	78.6%	21.4%	100.0%
			% within File	39.3%	17.6%	31.1%
		Yes	Count	17	14	31
			Expected Count	19.3	11.7	31.0
			% within Vulnerability Exploit	54.8%	45.2%	100.0%
			% within File	60.7%	82.4%	68.9%
	Total		Count	28	17	45
			Expected Count	28.0	17.0	45.0
			% within Vulnerability Exploit	62.2%	37.8%	100.0%
			% within File	100.0%	100.0%	100.0%

Chi-Square Tests

Email		Value	df	Asymp. Sig. (2-sided)	Exact Sig. (2-sided)	Exact Sig. (1-sided)
No	Pearson Chi-Square	11.923[b]	1	.001		
	Continuity Correction [a]	10.257	1	.001		
	Likelihood Ratio	11.908	1	.001		
	Fisher's Exact Test				.001	.001
	Linear-by-Linear Association	11.820	1	.001		
	N of Valid Cases	116				
Yes	Pearson Chi-Square	2.311[c]	1	.128		
	Continuity Correction [a]	1.412	1	.235		
	Likelihood Ratio	2.434	1	.119		
	Fisher's Exact Test				.188	.116
	Linear-by-Linear Association	2.260	1	.133		
	N of Valid Cases	45				

a. Computed only for a 2x2 table

b. 0 cells (.0%) have expected count less than 5. The minimum expected count is 8.10.

c. 0 cells (.0%) have expected count less than 5. The minimum expected count is 5.29.

Symmetric Measures

Email			Value	Approx. Sig.
No	Nominal by Nominal	Phi	-.321	.001
		Cramer's V	.321	.001
		Contingency Coefficient	.305	.001
	N of Valid Cases		116	
Yes	Nominal by Nominal	Phi	.227	.128
		Cramer's V	.227	.128
		Contingency Coefficient	.221	.128
	N of Valid Cases		45	

IV) Finding IV. Results for relationship between Vulnerability and Sharing directories.

Vulnerability Exploit * Sharing Directories Crosstabulation

			Sharing Directories		
			No	Yes	Total
Vulnerability Exploit	No	Count	67	43	110
		Expected Count	77.9	32.1	110.0
		% within Vulnerability Exploit	60.9%	39.1%	100.0%
		% within Sharing Directories	58.8%	91.5%	68.3%
	Yes	Count	47	4	51
		Expected Count	36.1	14.9	51.0
		% within Vulnerability Exploit	92.2%	7.8%	100.0%
		% within Sharing Directories	41.2%	8.5%	31.7%
Total		Count	114	47	161
		Expected Count	114.0	47.0	161.0
		% within Vulnerability Exploit	70.8%	29.2%	100.0%
		% within Sharing Directories	100.0%	100.0%	100.0%

Chi-Square Tests

	Value	df	Asymp. Sig. (2-sided)	Exact Sig. (2-sided)	Exact Sig. (1-sided)
Pearson Chi-Square	16.460[b]	1	.000		
Continuity Correction [a]	14.983	1	.000		
Likelihood Ratio	19.189	1	.000		
Fisher's Exact Test				.000	.000
Linear-by-Linear Association	16.358	1	.000		
N of Valid Cases	161				

a. Computed only for a 2x2 table

b. 0 cells (.0%) have expected count less than 5. The minimum expected count is 14.89.

Symmetric Measures

		Value	Approx. Sig.
Nominal by Nominal	Phi	-.320	.000
	Cramer's V	.320	.000
	Contingency Coefficient	.305	.000
N of Valid Cases		161	

V) Finding V. Results for relationship between self activation and human trigger.

Self activation * Human Trigger Crosstabulation

			Human Trigger		Total
			No	Yes	
Self activation	No	Count	7	27	34
		Expected Count	20.5	13.5	34.0
		% within Self activation	20.6%	79.4%	100.0%
		% within Human Trigger	7.2%	42.2%	21.1%
	Yes	Count	90	37	127
		Expected Count	76.5	50.5	127.0
		% within Self activation	70.9%	29.1%	100.0%
		% within Human Trigger	92.8%	57.8%	78.9%
Total		Count	97	64	161
		Expected Count	97.0	64.0	161.0
		% within Self activation	60.2%	39.8%	100.0%
		% within Human Trigger	100.0%	100.0%	100.0%

Chi-Square Tests

	Value	df	Asymp. Sig. (2-sided)	Exact Sig. (2-sided)	Exact Sig. (1-sided)
Pearson Chi-Square	28.308[b]	1	.000		
Continuity Correction[a]	26.248	1	.000		
Likelihood Ratio	28.557	1	.000		
Fisher's Exact Test				.000	.000
Linear-by-Linear Association	28.132	1	.000		
N of Valid Cases	161				

a. Computed only for a 2x2 table

b. 0 cells (.0%) have expected count less than 5. The minimum expected count is 13.52.

Symmetric Measures

		Value	Approx. Sig.
Nominal by Nominal	Phi	-.419	.000
	Cramer's V	.419	.000
	Contingency Coefficient	.387	.000
N of Valid Cases		161	

VI) Finding VI. Results for relationship between autorun registry and backdoor.

Autorun at registry * Backdoor Crosstabulation

			Backdoor		Total
			No	Yes	
Autorun at registry	No	Count	77	11	88
		Expected Count	70.5	17.5	88.0
		% within Autorun at registry	87.5%	12.5%	100.0%
	Yes	Count	52	21	73
		Expected Count	58.5	14.5	73.0
		% within Autorun at registry	71.2%	28.8%	100.0%
Total		Count	129	32	161
		Expected Count	129.0	32.0	161.0
		% within Autorun at registry	80.1%	19.9%	100.0%

Chi-Square Tests

	Value	df	Asymp. Sig. (2-sided)	Exact Sig. (2-sided)	Exact Sig. (1-sided)
Pearson Chi-Square	6.630[b]	1	.010		
Continuity Correction[a]	5.648	1	.017		
Likelihood Ratio	6.654	1	.010		
Fisher's Exact Test				.016	.009
Linear-by-Linear Association	6.589	1	.010		
N of Valid Cases	161				

a. Computed only for a 2x2 table

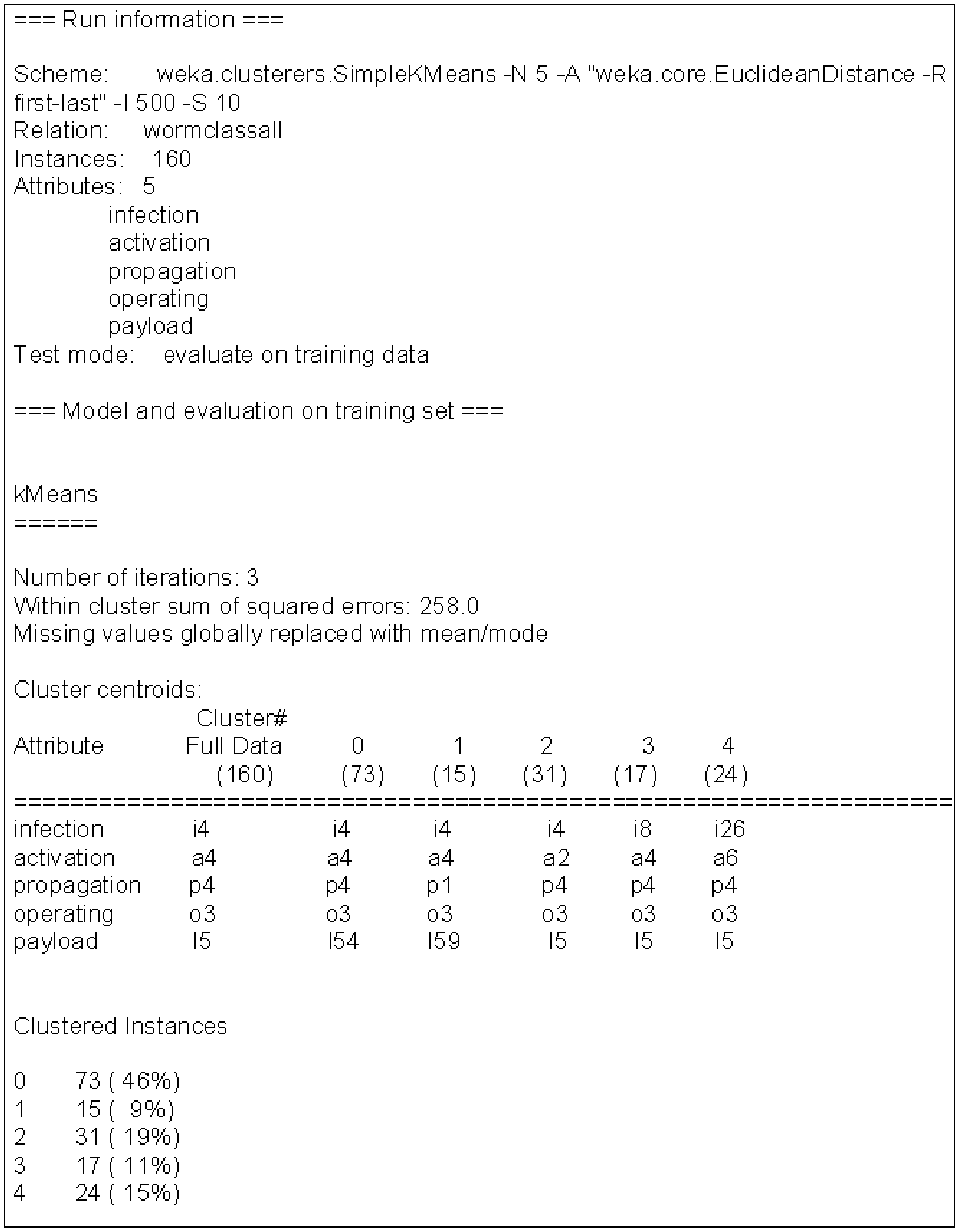
b. 0 cells (.0%) have expected count less than 5. The minimum expected count is 14.51.

Symmetric Measures

		Value	Approx. Sig.
Nominal by Nominal	Phi	.203	.010
	Cramer's V	.203	.010
	Contingency Coefficient	.199	.010
N of Valid Cases		161	

APPENDIX B:
CLUSTERING AND DETAILS ON WORM TYPE

I) WEKA clustering results



```
=== Run information ===

Scheme:      weka.clusterers.SimpleKMeans -N 5 -A "weka.core.EuclideanDistance -R
first-last" -I 500 -S 10
Relation:    wormclassall
Instances:   160
Attributes:  5
             infection
             activation
             propagation
             operating
             payload
Test mode:   evaluate on training data

=== Model and evaluation on training set ===


kMeans
======

Number of iterations: 3
Within cluster sum of squared errors: 258.0
Missing values globally replaced with mean/mode

Cluster centroids:
                    Cluster#
Attribute          Full Data      0        1       2       3      4
                     (160)       (73)     (15)    (31)    (17)   (24)
======================================================================
infection          i4            i4       i4       i4      i8     i26
activation         a4            a4       a4       a2      a4     a6
propagation        p4            p4       p1       p4      p4     p4
operating          o3            o3       o3       o3      o3     o3
payload            l5            l54      l59      l5      l5     l5


Clustered Instances

0       73 ( 46%)
1       15 (  9%)
2       31 ( 19%)
3       17 ( 11%)
4       24 ( 15%)
```

**i4 = file, i8 = sharing directories, i26 = email and vulnerability.*
a2 = human trigger, a4 = self activation, a6 = human trigger and self activation.
p1 = random, p4 = none.o3 = TSR.
l5= destruction, l54 = degrade performance, l59 = backdoor and autorun registry.

II) Details on the worm group type

Infection	Activation	Propagation	Operating Algorithm	Payload	Worm Type
vulnerability	self activation	none	TSR	worm generator tool & autorun registry	worm1
file & sharing directories	self activation	none	polymorphic	download file from website & compress, append & encrypt	worm1
file & sharing directories	self activation	none	TSR	mass mailing & autorun registry	worm1
email	no activation	none	TSR	backdoor & autorun registry	worm1
file	self activation	none	TSR	infect microsoft office, autorun registry & create infected file	worm1
USB	self activation	none	TSR	autorun registry	worm1
file	no activation	none	TSR	worm generator tool & autorun registry	worm1
chat	self activation	none	TSR	autorun registry	worm1
file	scheduled process	none	TSR	modify system.ini & format hard disk	worm1
file	human trigger & self activation	none	stealth	autorun registry, infect PE executable, send spam via chatting channel & format hard disk	worm1
vulnerability	self activation	none	TSR	command & control, autorun registry, infect PE executable, forward info to attacker, display message, kill certain processes, open port	worm1
file & sharing directories	no activation	none	TSR	worm generator tool & autorun registry	worm1
file & email	no activation	none	TSR	mass mailing, autorun registry, infect PE executable & hijack web browser	worm1
file	human trigger, scheduled process & self activation	none	Stealth	destruction, command & control, autorun registry, infect PE executable, display message, steal password, privilege escalated	worm1
file, sharing directories & chat	no activation	none	TSR	download file from website, autorun registry, upload/download file, forward info to attacker, infect autoexec.bat	worm1
file & vulnerability	self activation	sequence	TSR	destruction, command & control, download file from website	worm1
file, website & P2P	self activation	none	TSR	command & control	worm1
file & sharing directories	self activation	none	TSR	backdoor, mass mailing, infect local & removable drives, infect PE executable, forward info to attacker, create infected file, disable security protection	worm1
file	self activation	none	TSR	create infected .exe file	worm1

file	self activation	none	TSR	backdoor	worm1
file	human trigger & self activation	none	TSR	autorun registry & create infected .exe file	worm1
file	human trigger & self activation	none	TSR	backdoor, download file from website & infect PE executable	worm1
file	human trigger & self activation	none	TSR	backdoor, DOS, autorun registry, infect PE executable, steal password, privilege access escalated & create infected .exe file	worm1
vulnerability	self activation	sequence	TSR	backdoor, autorun registry, upload/download file & display message	worm1
email & vulnerability	self activation	none	TSR	destruction, display message, disable security protection	worm1
sharing directories & vulnerability	self activation	none	TSR	command & control, create infected.exe file & startup with .pif or .exe file	worm1
email & vulnerability	self activation	none	TSR	command & control, infect microsoft office & create, delete or corrupt file	worm1
website	self activation	none	TSR	backdoor, autorun registry, forward info to attacker, steal password & disable security protection	worm1
vulnerability	self activation	none	TSR	autorun registry	worm1
file	human trigger & self activation	none	TSR	infect PE executable, scan network & display message	worm1
floppy & USB	self activation	none	TSR	destruction & infect local & removable drives	worm1
chat	self activation	none	TSR	autorun registry, create infected .exe file & disable security protection	worm1
sharing directories & P2P	self activation	none	TSR	autorun registry & create infected .exe file	worm1
email & vulnerability	self activation	none	TSR	steal online banking info	worm1
file	self activation	none	TSR	mass mailing, autorun registry, forward info to attacker, steal password, create infected .exe file, rename .exe with other	worm1
file	self activation	none	TSR	mass mailing, autorun registry, forward info to attacker, steal password, create infected .exe file, rename .exe with other name	worm1
sharing directories & vulnerability	self activation	none	TSR	destruction, autorun registry, privilege access escalated & create, delete or corrupt file	worm1
P2P & chat	self activation	none	TSR	forward info to attacker & disable security protection	worm1
vulnerability	self activation	none	TSR	autorun registry, display message, create infected .exe file, startup with .pif and .exe file, infect HTML file& disable security protection	worm1

email, sharing directories, P2P & chat	self activation	none	TSR	command & control, mass mailing,autorun registry, modify system.ini, create infected .exe file & startup with .pif or .exe file	worm1
email & sharing directories	self activation	none	TSR	mass mailing & forward info to attacker	worm1
email	scheduled process & self activation	sequence	TSR	autorun registry, infect PE executable, infect link on dekstop, scan network, display message, rename .exe with other & infect HTML file	worm1
file	self activation	none	TSR	backdoor	worm1
file	self activation	none	TSR	infect microsoft office, autorun registry, display message & create infected .exe file	worm1
file & sharing directories	self activation	none	TSR	backdoor, autorun registry, infect local & removable drives, infect PE executable & open port	worm1
file, email & vulnerability	self activation	none	TSR	degrade performance	worm1
file	self activation	none	TSR	destruction & degrade performance	worm1
file, email & vulnerability	self activation	none	TSR	degrade performance	worm1
file, email & vulnerability	self activation	none	TSR	degrade performance	worm1
file	self activation	none	TSR	degrade performance	worm1
file	self activation	none	TSR	degrade performance	worm1
file	self activation	none	TSR	degrade performance	worm1
file	self activation	none	TSR	autorun registry, create infected .exe file & degrade performance	worm1
file	self activation	none	TSR	autorun registry, infect PE executable & display message	worm1
file, email & vulnerability	self activation	none	TSR	degrade performance	worm1
sharing directories	self activation	none	TSR	degrade performance	worm1
file	self activation	none	TSR	degrade performance	worm1
file, email & vulnerability	self activation	none	TSR	autorun registry & degrade performance	worm1
file & sharing directories	self activation	none	TSR	autorun registry, infect local & removable drives& create infected .exe file	worm1
file, email & vulnerability	self activation	none	TSR	degrade performance	worm1
file, email & vulnerability	self activation	none	TSR	degrade performance	worm1
file, email & vulnerability	self activation	none	TSR	autorun registry, infect PE executable & degrade performance	worm1
file, email & vulnerability	self activation	none	TSR	degrade performance	worm1
email & sharing	self activation	none	TSR	forward info to attacker	worm1

email	self activation	none	TSR	mass mailing, autorun registry, deny access to security website, display message, kill certain processes, create infected .exe file, rename .exe with other & disable security protection	worm1
CD	self activation	sequence	TSR	download file from website, infect PE executable, upload & download file & disable security protection	worm1
file & sharing directories	self activation	none	TSR	DOS, autorun registry, delete file in writable drives, kill certain processes, steal password, create infected .exe file, create &delete & corrupt file	worm1
file	self activation	none	TSR	backdoor, autorun registry & infect local & removable drives	worm1
file, email & vulnerability	self activation	none	TSR	degrade performance	worm1
file	self activation	none	TSR	worm generator tool & autorun registry	worm1
file	self activation	none	TSR	autorun registry & infect PE executable	worm1
file, email & vulnerability	self activation	none	TSR	degrade performance	worm1
file, email & vulnerability	self activation	none	TSR	degrade performance	worm1
file, email & sharing	self activation	random	TSR	backdoor, DOS, destruction, autorun registry & disable security protection	worm2
sharing	self activation	random	TSR	backdoor & autorun registry	worm2
file	self activation	random	TSR	backdoor, destruction, command & control & autorun registry	worm2
vulnerability	self activation	random	TSR	DOS, command & control & autorun registry	worm2
file	human trigger	random	TSR	backdoor & autorun registry	worm2
file & sharing directories	human trigger & self activation	random	TSR	exploit vulnerability based on OS version, infect PE executable & infect link on desktop	worm2
sharing directories	self activation	random	TSR	DOS, command & control, upload file	worm2
file	human trigger & self activation	random	TSR	backdoor, command & control, autorun registry & privilege access escalated	worm2
sharing directories & vulnerability	human trigger & self activation	random	TSR	backdoor, destruction, command & control, display message, steal password & privilege access escalated	worm2
sharing directories	human trigger & self activation	random	Stealth	scan network, format hard disk, startup with .pif or .exe file & autowar dialer	worm2
vulnerability & chat	self activation	random	anti anti-virus	command & control, download file from website, autorun registry & create, delete & corrupt file	worm2

sharing directories & chat	human trigger & self activation	random	TSR	destruction, autorun registry, privilege access escalated & create, delete or corrupt file	worm2
file & P2P	self activation	random	TSR	autorun registry, infect PE executable & print garbage	worm2
file & sharing directories	self activation	random	TSR	command & control, autorun registry, scan network, privilege access escalated & create infected .exe file	worm2
file	self activation	random	TSR	backdoor, DOS, autorun registry, open port & create infected .exe file	worm2
email	human trigger	none	TSR	mass mailing	worm3
file	human trigger	none	TSR	backdoor, command & control & autorun registry	worm3
file	human trigger	none	TSR	display message	worm3
file	human trigger	none	TSR	destruction	worm3
file & sharing directories	human trigger	none	TSR	autorun registry	worm3
File	human trigger	none	TSR	autorun registry	worm3
USB & file	human trigger	none	TSR	autorun registry, infect local & removable drives & infect PE executable	worm3
chat	human trigger	none	polymorphic	infect local & removable drives	worm3
file	human trigger	none	TSR	infect local & removable drives	worm3
chat	human trigger	none	TSR	autorun registry & display message	worm3
floppy	human trigger	none	TSR	autorun registry	worm3
email & sharing directories	human trigger	none	TSR	backdoor, command & control, infect autoexec.bat & create infected .exe file	worm3
file	human trigger	none	TSR	autorun registry	worm3
floppy	human trigger	none	TSR	autorun registry, redirect PC ports, display message & create infected .exe file	worm3
file & sharing directories	human trigger	none	TSR	autorun registry, infect PE executable, display message & privilege access escalated	worm3
file & vulnerability	human trigger	none	TSR	autorun registry, kill certain processes, delete registry with security & startup with .pif or .exe file	worm3
floppy & file	human trigger	none	TSR	autorun registry, infect PE executable & disguises as flash animation	worm3
file	human trigger	none	TSR	create infected .exe file & startup with .pif and .exe file	worm3
email & vulnerability	human trigger	none	TSR	destruction	worm3
sharing directories	human trigger	none	TSR	mass mailing & create infected .exe file	worm3

file	human trigger	none	TSR	backdoor, autorun registry, forward info to attacker, delete/ammend win.ini, modify system.ini & create infected .exe file	worm3
file	human trigger	none	TSR	backdoor, compress, append & encrypt, create infected file, steal password & create infected .exe file	worm3
floppy	human trigger	none	TSR	autorun registry, forward info to attacker, create infected .exe file & startup with .pif or .exe file	worm3
file	human trigger	none	TSR	destruction	worm3
file	human trigger	none	TSR	backdoor & infect local & removable drives	worm3
file	human trigger	none	TSR	display message, create infected .exe file & disguises as flash animation	worm3
file	human trigger	none	TSR	destruction	worm3
vulnerability & chat	human trigger	none	TSR	command & control, autorun registry, infect PE executable, open port & create infected .exe file	worm3
file	human trigger	none	TSR	worm generator tool & autorun registry	worm3
floppy & file	human trigger	none	TSR	delete/ammend win.ini, delete file in writable drives & display message	worm3
chat	human trigger	none	TSR	backdoor & infect local & removable drives	worm3
sharing directories	self activation	none	TSR	backdoor, destruction & autorun registry	worm4
sharing directories	no activation	none	TSR	backdoor, download file from website, autorun registry, deny access to security website & disable security protection	worm4
sharing directories	self activation	none	TSR	backdoor, autorun registry & delete network drives	worm4
email & vulnerability	self activation	none	TSR	destruction	worm4
email & vulnerability	self activation	sequence	TSR	destruction	worm4
email & vulnerability	self activation	none	TSR	destruction	worm4
email & vulnerability	self activation	none	TSR	destruction	worm4
sharing directories	self activation	none	TSR	destruction	worm4
email & vulnerability	self activation	none	TSR	destruction	worm4
sharing directories	human trigger & self activation	none	TSR	create log file capture malicious activity, infect local & removable drives & create infected .exe file	worm4
sharing directories	self activation	none	TSR	backdoor, DOS, command & control, reboot or log off, autorun registry, infect PE executable, upload/download file, forward info to attacker & create infected .exe file	worm4
sharing	self	random	TSR	destruction	worm4

directories	activation				
sharing directories	self activation	none	TSR	backdoor, DOS, command & control, reboot or log off, autorun registry, forward info to attacker & create infected .exe file	worm4
file & vulnerability	self activation	none	TSR	destruction	worm4
file, email & vulnerability	self activation	none	TSR	destruction	worm4
file, email & vulnerability	self activation	none	TSR	destruction	worm4
sharing directories	self activation	none	TSR	autorun registry	worm4
vulnerability	human trigger & self activation	none	TSR	destruction	worm5
email & vulnerability	human trigger & self activation	none	TSR	destruction	worm5
email, website & vulnerability	human trigger & self activation	none	TSR	backdoor, destruction & infect local & removable drives	worm5
file & vulnerability	human trigger & self activation	none	stealth	DOS, upload/download file, forward info to attacker, display message & disable security protection	worm5
website, vulnerability & chat	human trigger & self activation	none	TSR	download file from website, autorun registry & upload /download file	worm5
file, sharing directories & chat	human trigger & self activation	none	TSR	backdoor, DOS, download file from website, autorun registry, infect PE executable, send spam via chatting channel, redirect PC ports, scan network & display message	worm5
file & vulnerability	human trigger & self activation	none	Stealth	autorun registry, infect PE executable, forward info to attacker & open port	worm5
sharing directories & vulnerability	human trigger & self activation	none	TSR	install hacker's tool, backdoor, command & control, upload/download file, install spyware & steal password	worm5
file & sharing directories	human trigger & self activation	none	TSR	autorun registry & infect local & removable drives	worm5
file & sharing directories	human trigger & self activation	none	TSR	infect microsoft office, autorun registry & & infect PE executable	worm5
file, email & sharing directories	human trigger & self activation	none	TSR	delete registry with security & disable security protection	worm5
chat	human trigger & self activation	none	TSR	download file from website	worm5
file & sharing directories	human trigger & self activation	sequence	TSR	delete/ammend win.ini	worm5

email & vulnerability	human trigger & self activation	none	TSR	destruction	worm5
email & vulnerability	human trigger & self activation	none	TSR	destruction	worm5
email & vulnerability	human trigger & self activation	none	TSR	destruction	worm5
email & vulnerability	human trigger & self activation	none	TSR	destruction	worm5
floppy, USB & sharing directories	human trigger & self activation	none	TSR	infect local & removable drives, infect PE executable & infect link on desktop	worm5
email, sharing directories, P2P & chat	human trigger & self activation	none	TSR	DOS, infect PE executable, delete file in writable drives & create infected .exe file	worm5
email & P2P	human trigger & self activation	none	TSR	autorun registry, infect PE executable, display message, create infected .exe file & create ,delete or corrupt file	worm5
sharing directories & vulnerability	human trigger & self activation	none	TSR	autorun registry, infect local & removable drives& create infected .exe file	worm5
email & vulnerability	human trigger & self activation	none	TSR	destruction	worm5
email & vulnerability	human trigger & self activation	none	TSR	destruction	worm5
smartphone	human trigger & self activation	none	TSR	autorun registry,infect PE executable & create infected .exe file	worm5

APPENDIX C: DETAILS OF STAKCERT RULES FOR ASSIGNING WEIGHT, SEVERITY AND APOPTOSIS

<u>Rules for weight assignment:</u>

1. ***If*** *payload* involves any backdoor activities, ***then*** the weight is *high.*

2. ***If*** *payload* involves any activities, which compromise security setting, ***then*** the weight is *high*.

3. ***If*** *payload* involves any activities, which install any new file or application by exploiting vulnerability in the user's computer and without the user's consent, ***then*** the weight is *high.*

4. ***If*** *payload* compromises the security setting, ***then*** the weight is *high.*

5. ***If*** *payload* opens an unrelated network connection, ***then*** the weight is *high.*

6. ***If*** *payload* involves polymorphic techniques for worm self-mutation, ***then*** the weight is *high.*

7. ***If*** *payload* involves any activities, which disable the security software or access to the security website, ***then*** the weight is *high.*

8. ***If*** *payload* involves any system file modification, ***then*** the weight is *high.*

9. ***If*** *payload* involves any malicious file installation, ***then*** the weight is *high.*

10. ***If*** *payload* involves any activities, which collect and transmit personal identifiable information either from the user's computer, remotely or via website, ***then*** the weight is *high.*

11. ***If*** *payload* involves any performance or stability degradation up to 80% from normal operation, ***then*** the weight is *high.*

12. ***If*** *payload* involves any activities, which disconnect the network or Internet connection, ***then*** the weight is *high.*

13. ***If*** *payload* involves any mass mailing feature, ***then*** the weight is *high.*

14. ***If*** *payload* involves any vulnerability exploitation, which is based on operating systems version, ***then*** the weight is *high.*

15. ***If*** *payload* involves any activities, which create or infect any executable file, ***then*** the weight is *high.*

16. ***If*** *payload* involves any activities of file deletion or alteration, ***then*** the weight is *high.*

17. ***If*** *payload* involves any activities, which escalate account privilege without administrator consent, ***then*** the weight is *high.*

18. ***If*** *payload* involves any activities which steal or compromise the password, ***then*** the weight is *high.*

19. ***If*** *payload* involves any activities listed in rules 1 to 18 either singly or in combination, ***then*** the weight is *high.*

20. ***If*** *payload* involves any malicious activities related with registry, ***then*** the weight is *medium.*

21. ***If*** *payload* involves any link to infect the user's computer, ***then*** the weight is *medium.*

22. ***If*** *payload* involves any network scanning or collecting any data from user's computer, ***then*** the weight is *medium.*

23. ***If*** *payload* involves any activities, which print any unrelated document, ***then*** the weight is *medium.*

24. ***If*** *payload* involves other than rules 1 to 19 and rules 25 to 28 and involves rule 20 to rule 23 either singly or in combination with a weight lower than medium, ***then*** the weight is *medium.*

25. ***If*** *payload* involves other program running, which did not bring any harm to the user's computer, ***then*** the weight is *low.*

26. ***If*** *payload* involves online habit tracking as displayed in the installed end user law agreement (EULA) software, ***then*** the weight is *low.*

27. ***If*** *payload* involves any activities, which do not involve or compromise any serious privacy risk or security setting, ***then*** the weight is *low.*

28. ***If*** *payload* involves any activities, which display advertisement messages, ***then*** the weight is *low.*

29. ***If*** *infection* involves the vulnerability of a file or email, ***then*** the weight is *high.*

30. ***If*** *infection* involves any combination featuring in rules 29 and 31, ***then*** the weight is *high.*

31. ***If*** *infection* involves website(s), sharing file(s) or directories, P2P, USB or chatting channel, ***then*** the weight is *medium.*

32. ***If*** *infection* involves other than rules 29 and 30 and involves rule 31 either singly or in combination with lower weight than medium, ***then*** the weight is *medium.*

33. ***If*** *infection* involves other than rules 29 to 32, ***then*** the weight is *low.*

34. ***If*** *activation* involves self activation or a hybrid launch, ***then*** the weight is *high.*

35. **If** *activation* involves rule 34 either singly or in combination with a weight lower than high, **then** the weight is *high.*

36. **If** *activation* involves a human trigger or scheduled process, **then** the weight is *medium.*

37. **If** *activation* involves rule 36 either singly or in combination with a weight lower than medium, **then** the weight is *medium.*

38. ***If*** *activation* involves other than rules 34 to 37, then the weight is low.

39. **If** *propagation* involves random or sequence, **then** the weight is *high.*

40. **If** *propagation* involves rule 39 either singly or in combination with a weight lower than high, **then** the weight is *high.*

41. **If** *propagation* involves passive, **then** the weight is *high.*

42. **If** *propagation* involves rule 41 either singly or in combination with a weight lower than medium, **then** the weight is *medium.*

43. ***If*** *propagation involves* other than rules 39 to 42, then the weight is low.

44. **If** *operating algorithm* involves polymorphic, stealth, or anti-virus, **then** the weight is *high.*

45. **If** *operating algorithm* involves rule 44 either singly or in combination with a weight lower than high, **then** the weight is *high.*

46. **If** *operating algorithm* involves terminate stay resident (TSR), **then** the weight is *medium.*

47. **If** *operating algorithm* involves rule 46 either singly or in combination with a weight lower than medium, **then** the weight is *medium.*

48. ***If*** *operating algorithm* involves other than rules 44 to 47, then the weight is low.

Next is the set of rules for assignment of severity and apoptosis. Earlier the weight was assigned for each of the five main attributes of a worm. Consequently, the severity and apoptosis rules were generated based on the weight assignment mentioned earlier.

<u>Rules for severity and apoptosis assignment:</u>

1. **If** the weight for the *payload* and *infection* is high, **then** the severity is *high* **and** triggers *apoptosis.*

2. **If** it involves the combination of rule 1 **and** the weight combination of the *propagation, activation* and *operating algorithm* is high, medium or low, **then** the severity is *high* **and** triggers *apoptosis.*

3. **If** the weight for the *payload* is medium and *infection* is high, **then** the severity is *high* **and** triggers *apoptosis.*

4. **If** it involves the combination of rule 3 **and** the weight combination of the *propagation, activation* and *operating algorithm* is high, medium or low, **then** the severity is *high* **and** triggers *apoptosis.*

5. **If** the weight for the *payload* is high and *infection* is medium, **then** the severity is *high* **and** triggers *apoptosis.*

6. **If** it involves the combination of rule 5 **and** the weight combination of the *propagation, activation* and *operating algorithm* is high, medium or low, **then** the severity is *high* **and** triggers *apoptosis.*

7. **If** the weight for the *payload* is high and *infection* is low, **then** the severity is *high* **and** triggers *apoptosis.*

8. **If** it involves the combination of rule 7 **and** the weight combination of the *propagation, activation* and *operating algorithm* is high, medium or low, **then** the severity is *high* **and** triggers *apoptosis.*

9. **If** the weight for the *payload* is low and *infection* is high, **then** the severity is *high* **and** triggers *apoptosis.*

10. **If** it involves the combination of rule 9 **and** the weight combination of the *propagation, activation* and *operating algorithm* are high, medium or low, **then** the severity is *high* **and** triggers *apoptosis.*

11. **If** the weight for the *payload* is medium and *infection* is medium, **then** the severity is *medium* **and** there is no apoptosis.

12. **If** the weight for the *payload* is low and *infection* is medium **and** the weight combination of the *propagation, activation* and *operating algorithm* is high, medium or low, **then** the severity is *medium* **and** there is no apoptosis.

13. **If** the weight for the *payload* is medium and *infection* is low **and** the weight combination of the *propagation, activation* and *operating algorithm* is high, medium or low, **then** the severity is *medium* **and** there is no apoptosis.

14. **If** the weight for the *payload* is low and *infection* is low **and** the weight combination of the *propagation, activation* and *operating algorithm* is high, **then** the severity is *medium* **and** there is no apoptosis.

15. If the weight for the *payload* is low and *infection* is low **and** the weight combination of the two attributes either from *propagation, activation* and *operating algorithm* is high **and** the other attributes weight is medium or low, **then** the severity is *medium* **and** there is no apoptosis.

16. **If** the weight for the *payload* is low and *infection* is low **and** the weight one of the attributes either from *propagation, activation* and *operating algorithm* is high **and** the other attributes weight is medium or low, **then** the severity is *medium* **and** there is no apoptosis.

17. **If** the weight for the *payload* is low and *infection* is low **and** the weight one of the attributes either from *propagation, activation* and *operating algorithm* is medium **and** the other attributes weight is low, **then** the severity is *medium* **and** there is no apoptosis.

18. **If** the weight for the *payload, activation, propagation, infection* and *operating algorithm* is low, **then** the severity is *low* **and** there is no apoptosis.

STAKCERT Worm Structural Pattern based on STAKCERT Weight, Severity and Apoptosis rules

No	**Payload**	**Infection**	**Activation**	**Propagation**	**Operating algorithm**	**Severity**
1	High	High	High	High	High	High
2	High	High	High	High	Medium	High
3	High	High	High	High	Low	High
4	High	High	High	Medium	High	High
5	High	High	High	Low	High	High
6	High	High	Medium	High	High	High
7	High	High	Low	High	High	High
8	High	Medium	High	High	High	High
9	High	Low	High	High	High	High
10	High	High	High	Medium	Medium	High
11	High	High	High	Low	Medium	High
12	High	High	Medium	High	Medium	High
13	High	High	Low	High	Medium	High
14	High	Medium	High	High	Medium	High
15	High	Low	High	High	Medium	High
16	High	High	Medium	Medium	Medium	High
17	High	High	Low	Medium	Medium	High
18	High	Medium	High	Medium	Medium	High
19	High	Low	High	Medium	Medium	High
20	High	Medium	Medium	Medium	Medium	High
21	High	Low	Medium	Medium	Medium	High
22	High	High	High	Medium	Low	High
23	High	High	High	Low	Low	High
24	High	High	Medium	High	Low	High
25	High	High	Low	High	Low	High
26	High	Medium	High	High	Low	High
27	High	Low	High	High	Low	High
28	High	High	Medium	Low	Low	High
29	High	High	Low	Low	Low	High
30	High	Medium	High	Low	Low	High
31	High	Low	High	Low	Low	High
32	High	Medium	Low	Low	Low	High
33	High	Low	Low	Low	Low	High
34	High	High	High	Low	High	High
35	High	High	High	Low	Medium	High
36	High	High	Medium	High	High	High
37	High	High	Low	High	High	High
38	High	High	Low	Low	Low	High
39	High	High	Low	Low	Medium	High
40	High	High	Low	Low	High	High
41	High	High	Low	Medium	High	High
42	High	High	Medium	Medium	Medium	High
43	High	High	Medium	Low	Medium	High
44	High	High	Medium	Low	High	High
45	High	High	Medium	Medium	High	High
46	High	Medium	High	High	High	High
47	High	Medium	High	High	Medium	High
48	High	Medium	High	High	Low	High
49	High	Medium	High	Medium	High	High
50	High	Medium	High	Low	High	High
51	High	Medium	High	Low	Medium	High
52	High	Medium	Medium	High	High	High

53	High	Medium	Low	High	High	High
54	High	Medium	Low	Low	Medium	High
55	High	Medium	Low	Low	High	High
56	High	Medium	Low	Medium	High	High
57	High	Medium	Medium	Medium	Medium	High
58	High	Medium	Medium	Low	Low	High
59	High	Medium	Medium	Low	Medium	High
60	High	Medium	Medium	Low	High	High
61	High	Medium	Medium	Medium	High	High
62	High	Medium	Medium	High	Medium	High
63	High	Low	High	High	High	High
64	High	Low	High	High	Medium	High
65	High	Low	High	High	Low	High
66	High	Low	High	Medium	High	High
67	High	Low	High	Low	High	High
68	High	Low	High	Low	Medium	High
69	High	Low	Medium	High	High	High
70	High	Low	Low	High	High	High
71	High	Low	Low	Low	Low	High
72	High	Low	Low	Low	Medium	High
73	High	Low	Low	Low	High	High
74	High	Low	Low	Medium	High	High
75	High	Low	Medium	Medium	Medium	High
76	High	Low	Medium	Low	Low	High
77	High	Low	Medium	Low	Medium	High
78	High	Low	Medium	Low	High	High
79	High	Low	Medium	Medium	High	High
80	Medium	High	High	High	High	High
81	Medium	High	High	High	Medium	High
82	Medium	High	High	Medium	Medium	High
83	Medium	High	Medium	Medium	Medium	High
84	Medium	Medium	Medium	Medium	Medium	Medium
85	Medium	High	High	High	Low	High
86	Medium	High	High	Low	Low	High
87	Medium	High	Low	Low	Low	High
88	Medium	Low	Low	Low	Low	Medium
89	Medium	High	High	High	High	High
90	Medium	High	High	High	Medium	High
91	Medium	High	High	High	Low	High
92	Medium	High	High	Medium	High	High
93	Medium	High	High	Low	High	High
94	Medium	High	Medium	High	High	High
95	Medium	High	Low	High	High	High
96	Medium	Medium	High	High	High	Medium
97	Medium	Medium	High	High	Medium	Medium
98	Medium	Medium	High	High	Low	Medium
99	Medium	Medium	High	Medium	High	Medium
100	Medium	Medium	High	Low	High	Medium
101	Medium	Medium	Medium	High	High	Medium
102	Medium	Medium	Low	High	High	Medium
103	Medium	Medium	Medium	High	High	Medium
104	Medium	Medium	Medium	High	Medium	Medium
105	Medium	Medium	Medium	High	Low	Medium
106	Medium	Medium	Medium	Medium	High	Medium
107	Medium	Medium	Medium	Low	High	Medium
108	Medium	Medium	Medium	Medium	High	Medium
109	Medium	Medium	Medium	Medium	Low	Medium
110	Medium	High	Medium	Low	High	High
111	Medium	High	Medium	High	High	High

112	Medium	Medium	Low	High	High	Medium
113	Medium	Medium	High	High	High	Medium
114	Medium	Low	High	High	High	Medium
115	Medium	High	High	High	High	High
116	Medium	Medium	Medium	Low	High	Medium
117	Medium	Medium	Medium	High	High	Medium
118	Medium	Medium	Low	High	High	Medium
119	Medium	Medium	High	High	High	Medium
120	Medium	Low	High	High	High	Medium
121	Medium	High	High	High	High	High
122	Medium	High	High	High	Medium	High
123	Medium	High	High	High	Low	High
124	Medium	High	High	Medium	High	High
125	Medium	High	High	Low	High	High
126	Medium	High	Medium	High	High	High
127	Medium	High	Low	High	High	High
128	Medium	High	Low	Low	Low	High
129	Medium	High	Low	Low	Medium	High
130	Medium	High	Low	Low	High	High
131	Medium	High	Low	Medium	High	High
132	Medium	High	Medium	Medium	Medium	High
133	Medium	High	Medium	Low	Low	High
134	Medium	High	Medium	Low	Medium	High
135	Medium	High	Medium	Low	High	High
136	Medium	High	Medium	Medium	High	High
137	Medium	Medium	High	High	High	Medium
138	Medium	Medium	High	High	Medium	Medium
139	Medium	Medium	High	High	Low	Medium
140	Medium	Medium	High	Medium	High	Medium
141	Medium	Medium	High	Low	High	Medium
142	Medium	Medium	Medium	High	High	Medium
143	Medium	Medium	Low	High	High	Medium
144	Medium	Medium	Low	Low	Low	Medium
145	Medium	Medium	Low	Low	Medium	Medium
146	Medium	Medium	Low	Low	High	Medium
147	Medium	Medium	Low	Medium	High	Medium
148	Medium	Medium	Medium	Medium	Medium	Medium
149	Medium	Medium	Medium	Low	Low	Medium
150	Medium	Medium	Medium	Low	Medium	Medium
151	Medium	Medium	Medium	Low	High	Medium
152	Medium	Medium	Medium	Medium	High	Medium
153	Medium	Low	High	High	High	Medium
154	Medium	Low	High	High	Medium	Medium
155	Medium	Low	High	High	Low	Medium
156	Medium	Low	High	Medium	High	Medium
157	Medium	Low	High	Low	High	Medium
158	Medium	Low	Medium	High	High	Medium
159	Medium	Low	Low	High	High	Medium
160	Medium	Low	Low	Low	Low	Medium
161	Medium	Low	Low	Low	Medium	Medium
162	Medium	Low	Low	Low	High	Medium
163	Medium	Low	Low	Medium	High	Medium
164	Medium	Low	Medium	Medium	Medium	Medium
165	Medium	Low	Medium	Low	Low	Medium
166	Medium	Low	Medium	Low	Medium	Medium
167	Medium	Low	Medium	Low	High	Medium
168	Medium	Low	Medium	Medium	High	Medium
169	Low	High	High	High	High	High
170	Low	High	High	High	Medium	High

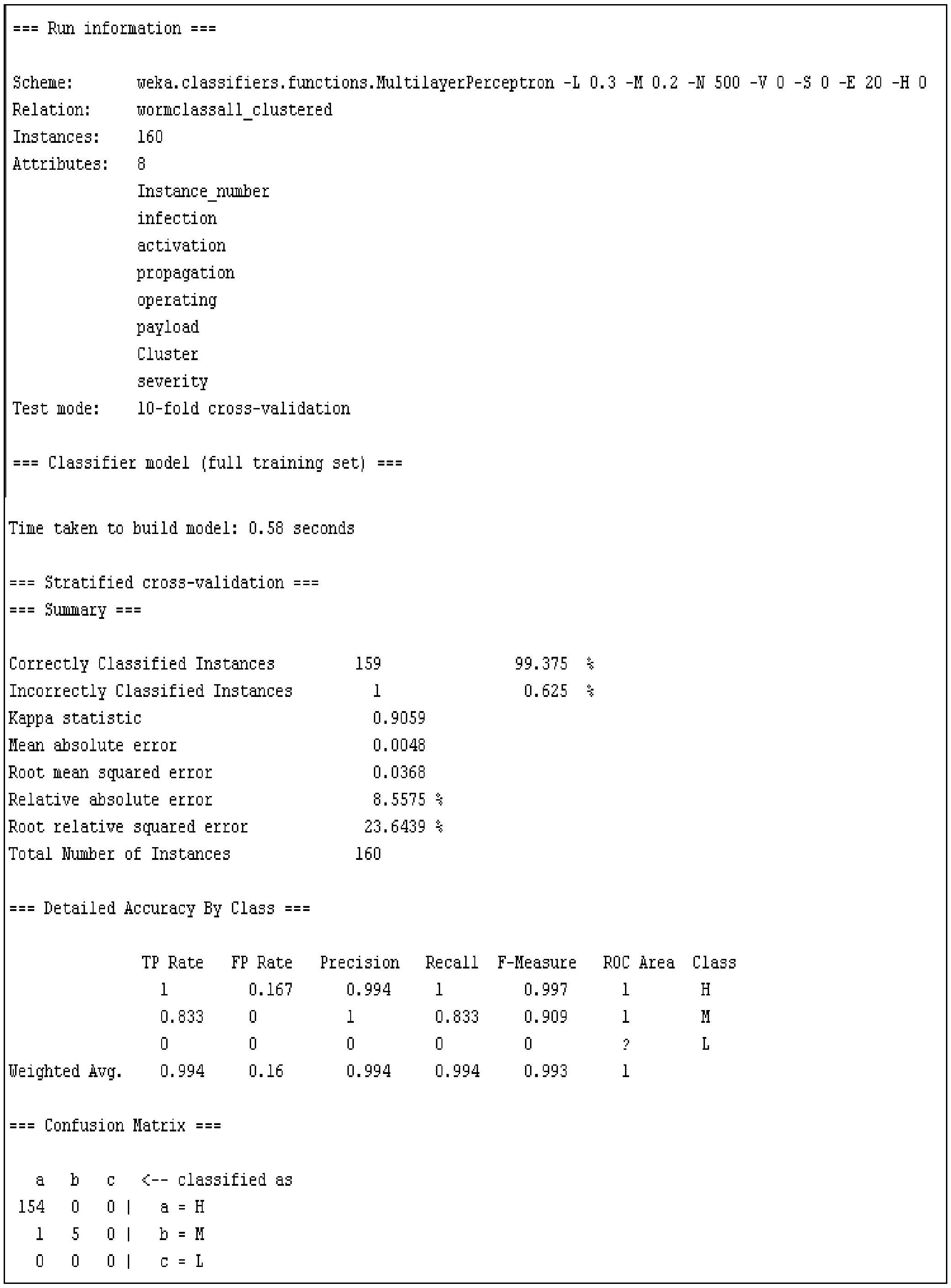
171	Low	High	High	Medium	Medium	High
172	Low	High	Medium	Medium	Medium	High
173	Low	Medium	Medium	Medium	Medium	Medium
174	Low	High	High	High	Low	High
175	Low	High	High	Low	Low	High
176	Low	High	Low	Low	Low	High
177	Low	High	High	High	High	High
178	Low	High	High	High	Medium	High
179	Low	High	High	High	Low	High
180	Low	High	High	Medium	High	High
181	Low	High	High	Low	High	High
182	Low	High	Medium	High	High	High
183	Low	High	Low	High	High	High
184	Low	Medium	High	High	High	Medium
185	Low	Low	High	High	High	Low
186	Low	Low	High	High	Medium	Low
187	Low	Low	High	High	Low	Low
188	Low	Low	High	Medium	High	Low
189	Low	Low	High	Low	High	Low
190	Low	Low	Medium	High	High	Low
191	Low	Low	Low	High	High	Low
192	Low	Low	Low	High	Medium	Low
193	Low	Low	Low	High	Low	Low
194	Low	Low	Low	Medium	High	Low
195	Low	Low	Low	Low	High	Low
196	Low	Low	Low	Low	Medium	Low
197	Low	Low	Low	Low	Low	Low
198	Low	High	High	High	High	High
199	Low	High	Medium	Low	High	High
200	Low	Medium	Low	High	High	Medium
201	Low	Medium	High	High	High	Medium
202	Low	Low	Medium	Low	High	Low
203	Low	Low	Low	High	High	Low
204	Low	High	High	High	High	High
205	Low	High	High	High	Medium	High
206	Low	High	High	High	Low	High
207	Low	High	High	Medium	High	High
208	Low	High	High	Low	High	High
209	Low	High	Medium	High	High	High
210	Low	High	Low	High	High	High
211	Low	High	Low	Low	Low	High
212	Low	High	Low	Low	Medium	High
213	Low	High	Low	Low	High	High
214	Low	High	Low	Medium	High	High
215	Low	High	Medium	Medium	Medium	High
216	Low	High	Medium	Low	Low	High
217	Low	High	Medium	Low	Medium	High
218	Low	High	Medium	Low	High	High
219	Low	High	Medium	Medium	High	High
220	Low	Medium	High	High	High	Medium
221	Low	Medium	High	High	Medium	Medium
222	Low	Medium	High	High	Low	Medium
223	Low	Medium	High	Medium	High	Medium
224	Low	Medium	High	Low	High	Medium
225	Low	Medium	Medium	High	High	Medium
226	Low	Medium	Low	High	High	Medium
227	Low	Medium	Medium	Medium	Medium	Medium
228	Low	Low	High	High	High	Low
229	Low	Low	High	High	Medium	Low

230	Low	Low	High	High	Low	Low
231	Low	Low	High	Medium	High	Low
232	Low	Low	High	Low	High	Low
233	Low	Low	Medium	High	High	Low
234	Low	Low	Low	High	High	Low
235	Low	Low	Low	Low	Low	Low
236	Low	Low	Low	Low	Medium	Low
237	Low	Low	Low	Low	High	Low
238	Low	Low	Low	Medium	High	Low
239	Low	Low	Medium	Medium	Medium	Low
240	Low	Low	Medium	Low	Low	Low
241	Low	Low	Medium	Low	Medium	Low
242	Low	Low	Medium	Low	High	Low
243	Low	Low	Medium	Medium	High	Low

Firstly, above table presents all the combinations of possible values for weight and severity. In this STAKCERT structural pattern table, the six main attributes are payload, infection, activation, propagation, operating algorithm, and severity. Each of these attributes has three classes: low, medium and high. The 243 rows presented are the three possible values for each attribute (3x3x3x3x3 =243). All of these patterns were generated based on the STAKCERT worm apoptosis rules.

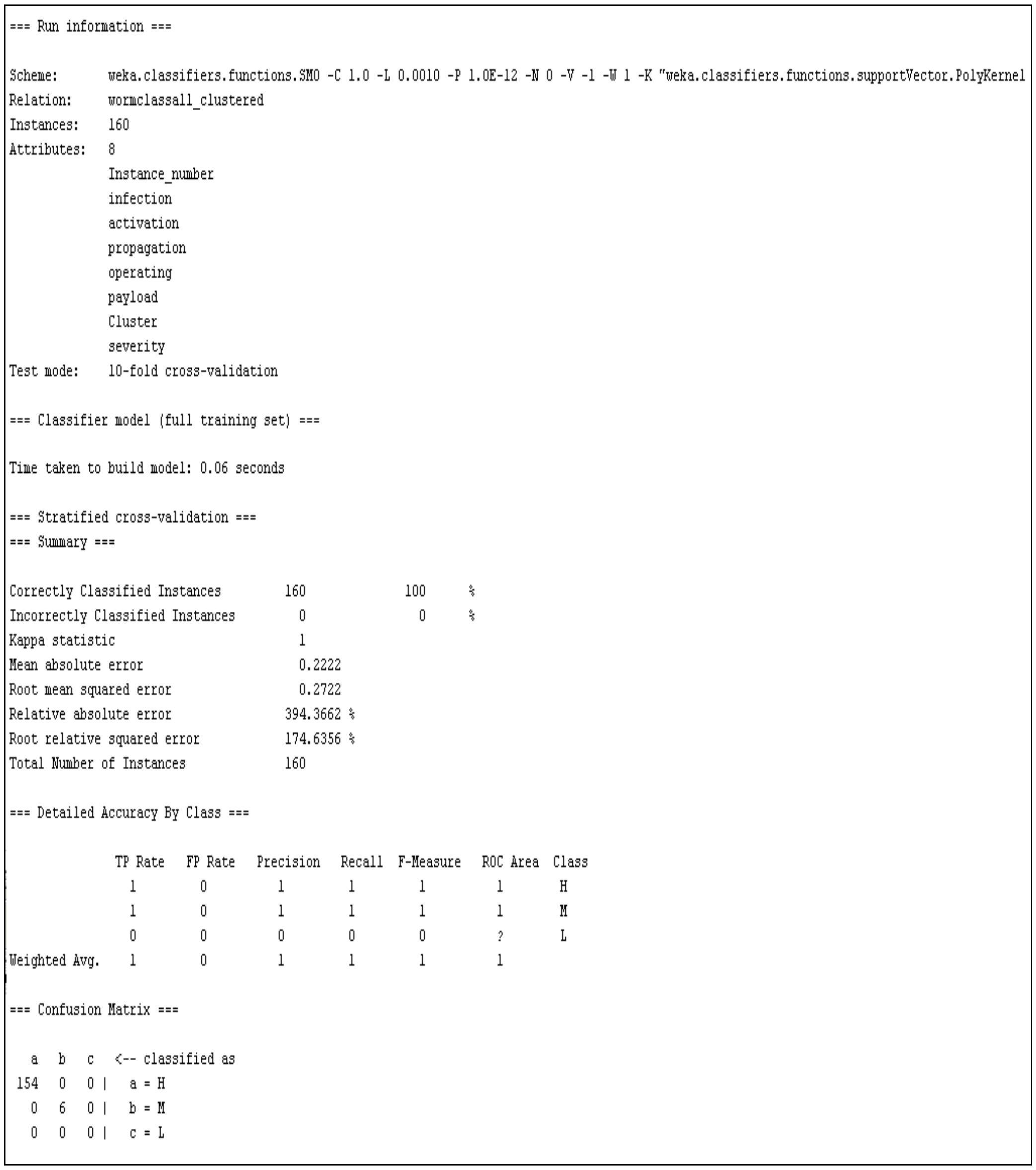
APPENDIX D:
STAKCERT WEIGHT AND SEVERITY RESULTS

I) Results for Multilayer Perceptron



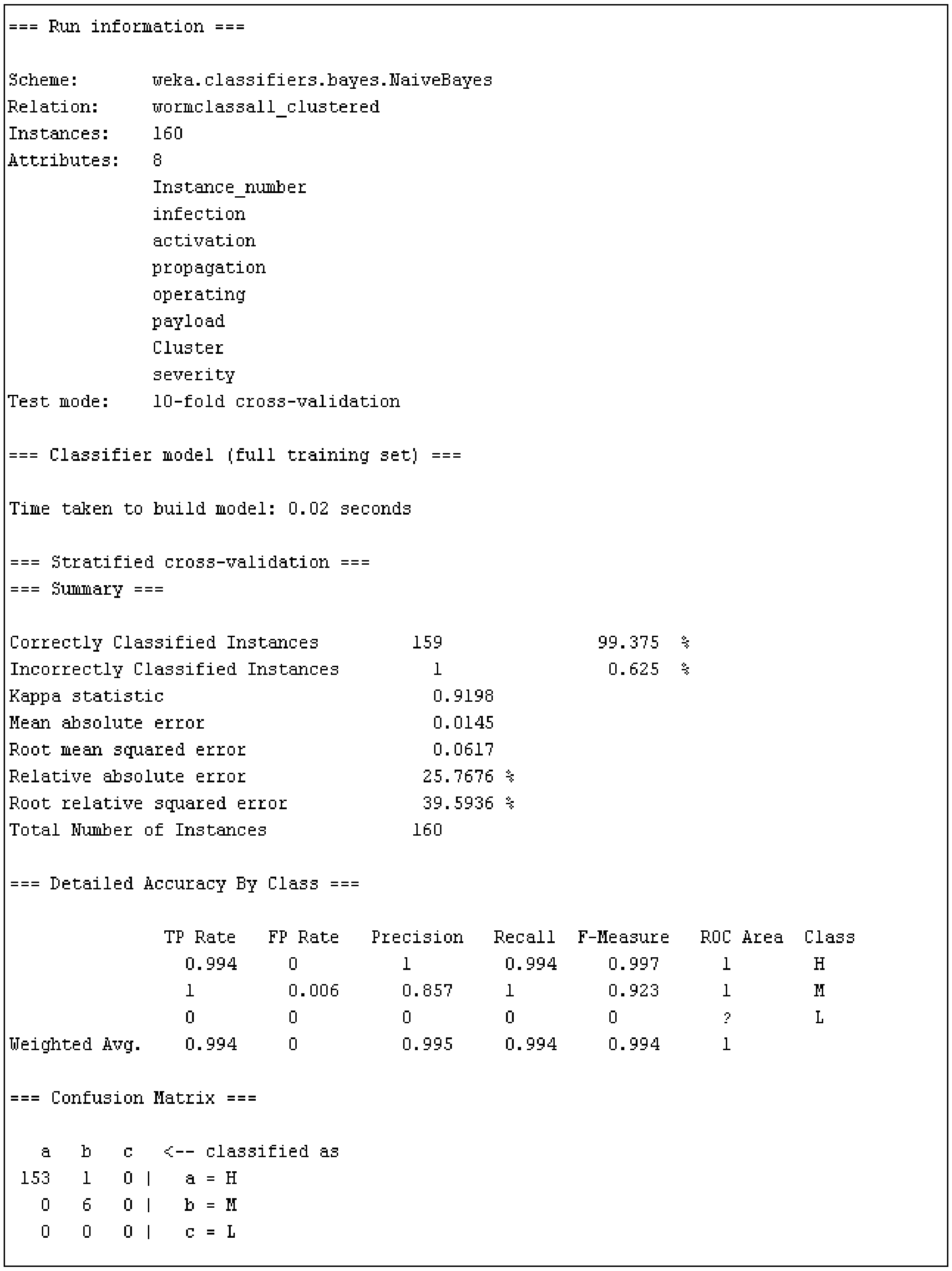
```
=== Run information ===

Scheme:       weka.classifiers.functions.MultilayerPerceptron -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H 0
Relation:     wormclassall_clustered
Instances:    160
Attributes:   8
              Instance_number
              infection
              activation
              propagation
              operating
              payload
              Cluster
              severity
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

Time taken to build model: 0.58 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        159               99.375  %
Incorrectly Classified Instances        1                0.625  %
Kappa statistic                         0.9059
Mean absolute error                     0.0048
Root mean squared error                 0.0368
Relative absolute error                 8.5575 %
Root relative squared error            23.6439 %
Total Number of Instances             160

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 1         0.167      0.994     1         0.997      1        H
                 0.833     0          1         0.833     0.909      1        M
                 0         0          0         0         0          ?        L
Weighted Avg.    0.994     0.16       0.994     0.994     0.993      1

=== Confusion Matrix ===

   a   b   c   <-- classified as
 154   0   0 |   a = H
   1   5   0 |   b = M
   0   0   0 |   c = L
```

II) Results for SMO Simulation



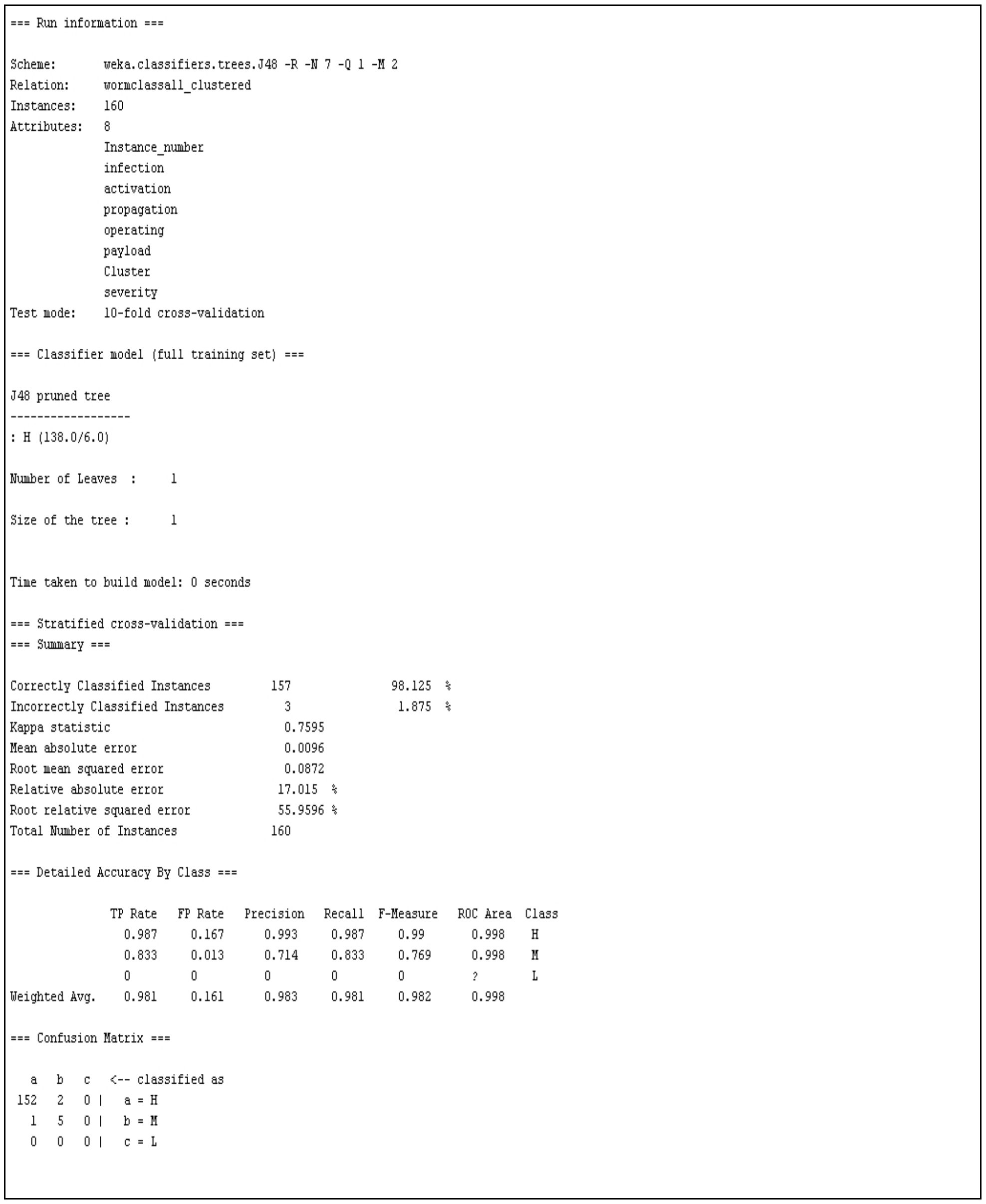
```
=== Run information ===

Scheme:       weka.classifiers.functions.SMO -C 1.0 -L 0.0010 -P 1.0E-12 -N 0 -V -1 -W 1 -K "weka.classifiers.functions.supportVector.PolyKernel
Relation:     wormclassall_clustered
Instances:    160
Attributes:   8
              Instance_number
              infection
              activation
              propagation
              operating
              payload
              Cluster
              severity
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

Time taken to build model: 0.06 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         160              100      %
Incorrectly Classified Instances         0                0      %
Kappa statistic                          1
Mean absolute error                      0.2222
Root mean squared error                  0.2722
Relative absolute error                394.3662 %
Root relative squared error            174.6356 %
Total Number of Instances              160

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 1         0          1          1         1          1        H
                 1         0          1          1         1          1        M
                 0         0          0          0         0          ?        L
Weighted Avg.    1         0          1          1         1          1

=== Confusion Matrix ===

   a   b   c   <-- classified as
 154   0   0 |   a = H
   0   6   0 |   b = M
   0   0   0 |   c = L
```

III) Results for Naïve Bayes



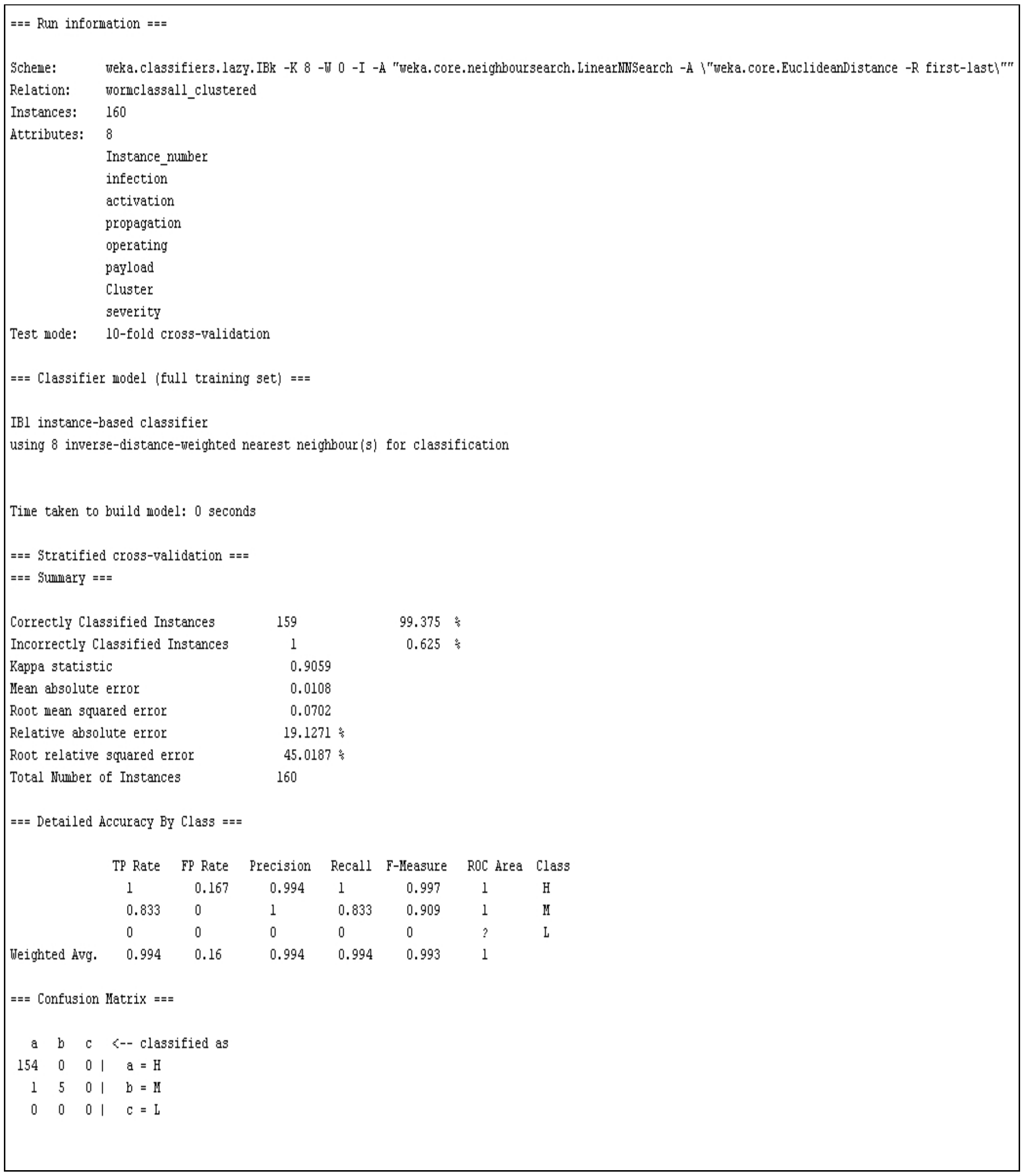
```
=== Run information ===

Scheme:       weka.classifiers.bayes.NaiveBayes
Relation:     wormclassall_clustered
Instances:    160
Attributes:   8
              Instance_number
              infection
              activation
              propagation
              operating
              payload
              Cluster
              severity
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

Time taken to build model: 0.02 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         159               99.375  %
Incorrectly Classified Instances         1                0.625  %
Kappa statistic                          0.9198
Mean absolute error                      0.0145
Root mean squared error                  0.0617
Relative absolute error                 25.7676 %
Root relative squared error             39.5936 %
Total Number of Instances              160

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.994     0          1         0.994     0.997      1        H
                 1         0.006      0.857     1         0.923      1        M
                 0         0          0         0         0          ?        L
Weighted Avg.    0.994     0          0.995     0.994     0.994      1

=== Confusion Matrix ===

   a   b   c   <-- classified as
 153   1   0 |   a = H
   0   6   0 |   b = M
   0   0   0 |   c = L
```

IV) Results for J48



```
=== Run information ===

Scheme:       weka.classifiers.trees.J48 -R -N 7 -Q 1 -M 2
Relation:     wormclassall_clustered
Instances:    160
Attributes:   8
              Instance_number
              infection
              activation
              propagation
              operating
              payload
              Cluster
              severity
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

J48 pruned tree
------------------
: H (138.0/6.0)

Number of Leaves  :     1

Size of the tree :      1


Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         157               98.125  %
Incorrectly Classified Instances         3                1.875  %
Kappa statistic                          0.7595
Mean absolute error                      0.0096
Root mean squared error                  0.0872
Relative absolute error                 17.015  %
Root relative squared error             55.9596 %
Total Number of Instances              160

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.987     0.167      0.993     0.987     0.99       0.998    H
                 0.833     0.013      0.714     0.833     0.769      0.998    M
                 0         0          0         0         0          ?        L
Weighted Avg.    0.981     0.161      0.983     0.981     0.982      0.998

=== Confusion Matrix ===

   a   b   c   <-- classified as
 152   2   0 |   a = H
   1   5   0 |   b = M
   0   0   0 |   c = L
```

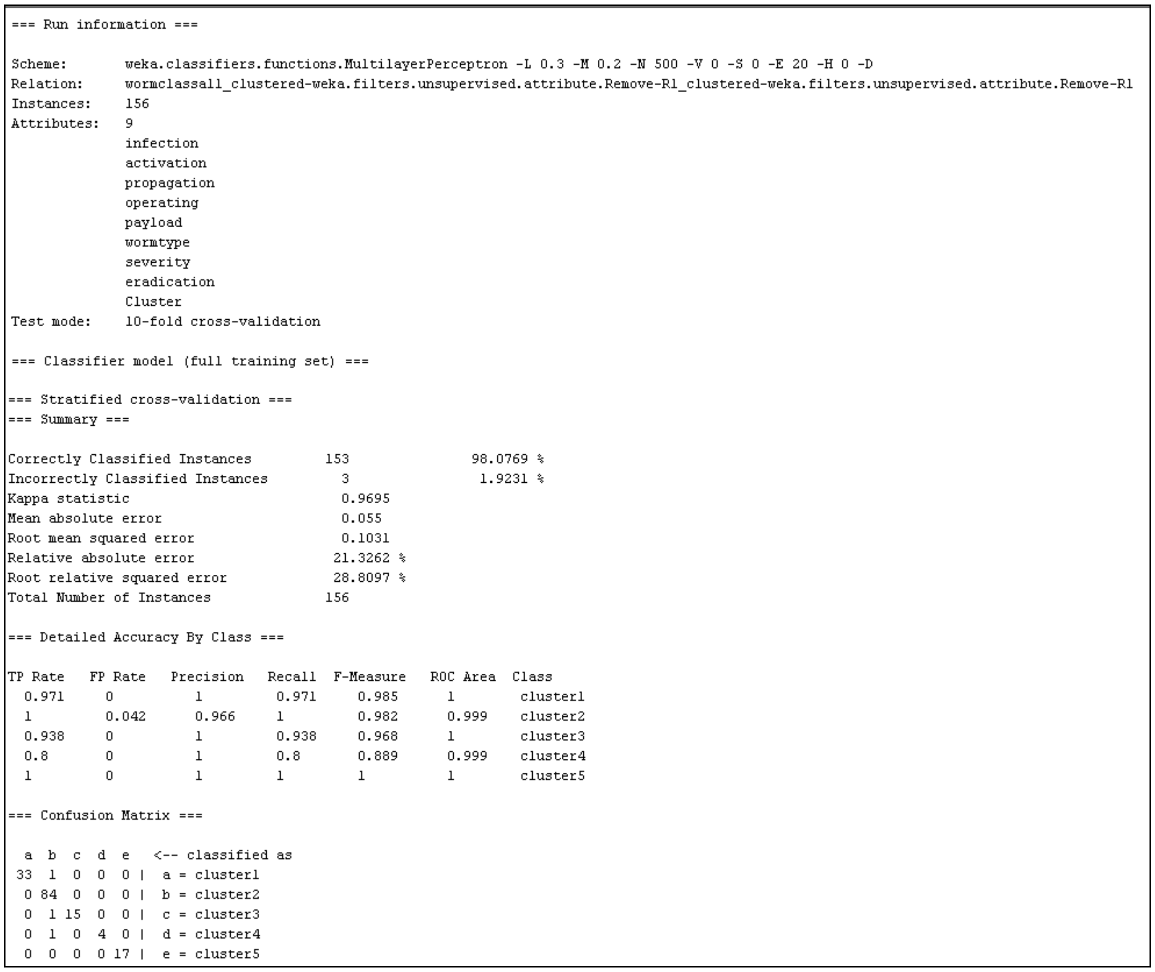
V) Results for IBk



```
=== Run information ===

Scheme:       weka.classifiers.lazy.IBk -K 8 -W 0 -I -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""
Relation:     wormclassall_clustered
Instances:    160
Attributes:   8
              Instance_number
              infection
              activation
              propagation
              operating
              payload
              Cluster
              severity
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

IB1 instance-based classifier
using 8 inverse-distance-weighted nearest neighbour(s) for classification


Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         159               99.375  %
Incorrectly Classified Instances         1                0.625  %
Kappa statistic                          0.9059
Mean absolute error                      0.0108
Root mean squared error                  0.0702
Relative absolute error                 19.1271 %
Root relative squared error             45.0187 %
Total Number of Instances              160

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 1         0.167     0.994     1         0.997      1        H
                 0.833     0         1         0.833     0.909      1        M
                 0         0         0         0         0          ?        L
Weighted Avg.    0.994     0.16      0.994     0.994     0.993      1

=== Confusion Matrix ===

   a   b   c   <-- classified as
 154   0   0 |   a = H
   1   5   0 |   b = M
   0   0   0 |   c = L
```

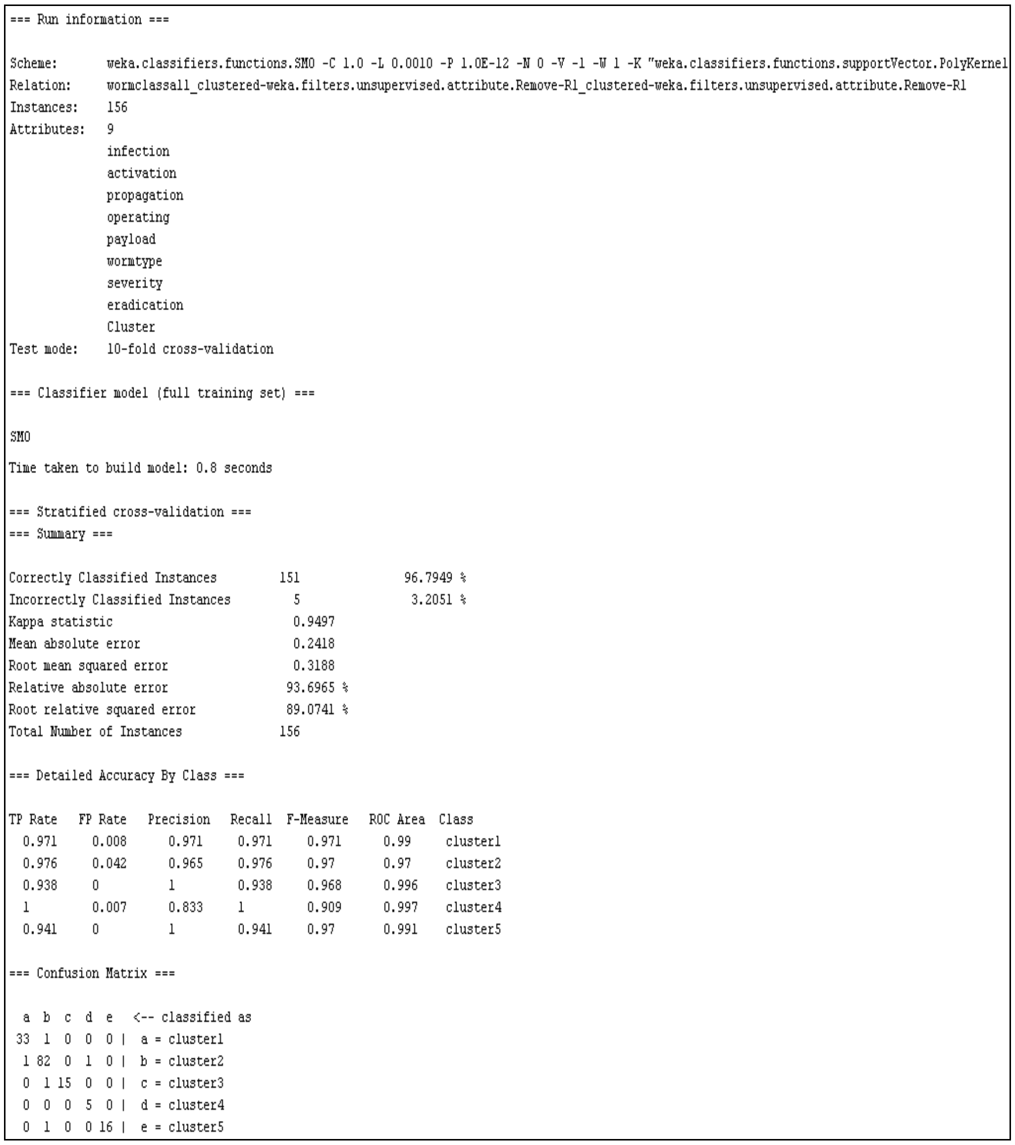
APPENDIX E: STAKCERT MODEL FOR WORM RESPONSE SIMULATION RESULTS

l) Results for Multilayer Perceptron Simulation



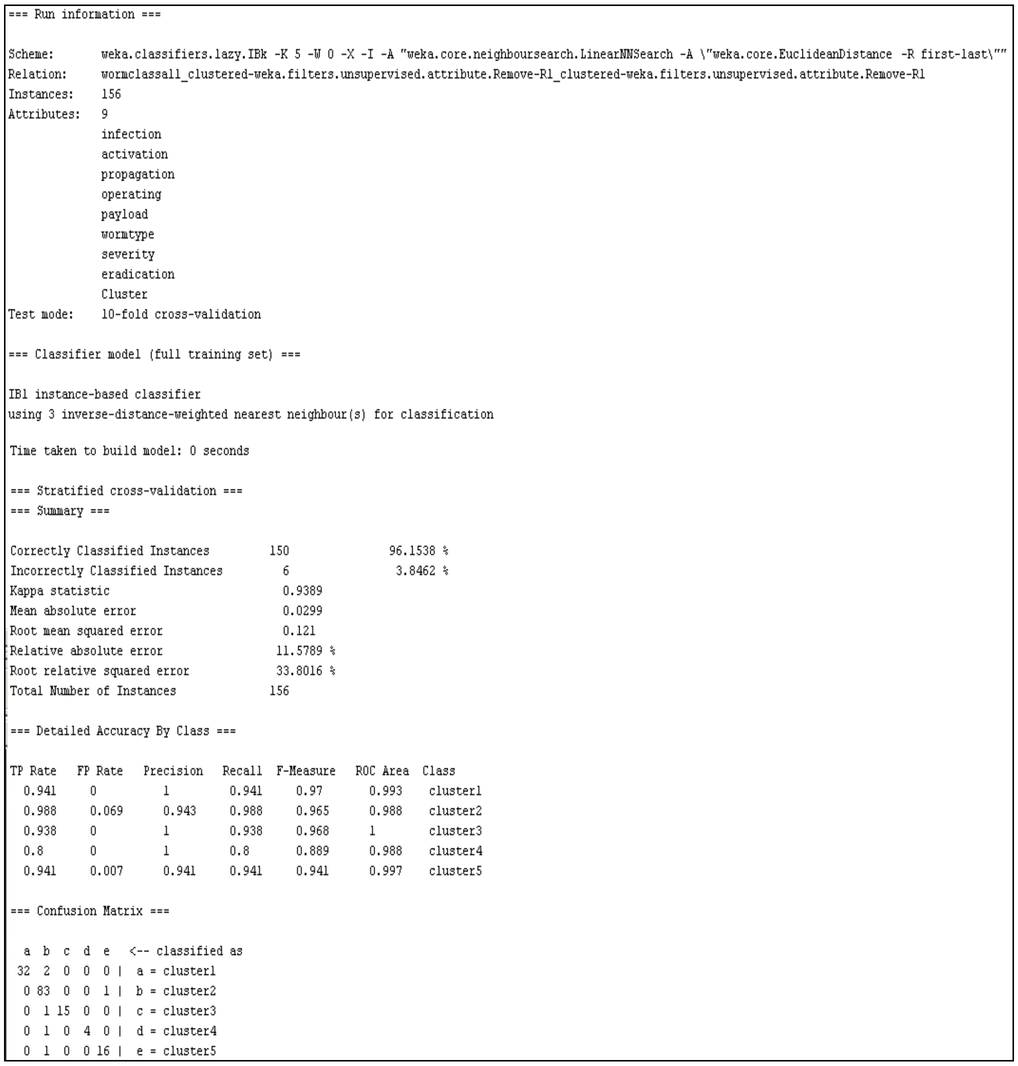
```
=== Run information ===

Scheme:       weka.classifiers.functions.MultilayerPerceptron -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H 0 -D
Relation:     wormclassall_clustered-weka.filters.unsupervised.attribute.Remove-R1_clustered-weka.filters.unsupervised.attribute.Remove-R1
Instances:    156
Attributes:   9
              infection
              activation
              propagation
              operating
              payload
              wormtype
              severity
              eradication
              Cluster
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         153               98.0769 %
Incorrectly Classified Instances         3                1.9231 %
Kappa statistic                          0.9695
Mean absolute error                      0.055
Root mean squared error                  0.1031
Relative absolute error                 21.3262 %
Root relative squared error             28.8097 %
Total Number of Instances              156

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
  0.971     0          1         0.971     0.985      1        cluster1
  1         0.042      0.966     1         0.982      0.999    cluster2
  0.938     0          1         0.938     0.968      1        cluster3
  0.8       0          1         0.8       0.889      0.999    cluster4
  1         0          1         1         1          1        cluster5

=== Confusion Matrix ===

  a  b  c  d  e   <-- classified as
 33  1  0  0  0 |  a = cluster1
  0 84  0  0  0 |  b = cluster2
  0  1 15  0  0 |  c = cluster3
  0  1  0  4  0 |  d = cluster4
  0  0  0  0 17 |  e = cluster5
```

II) Results for SMO Simulation



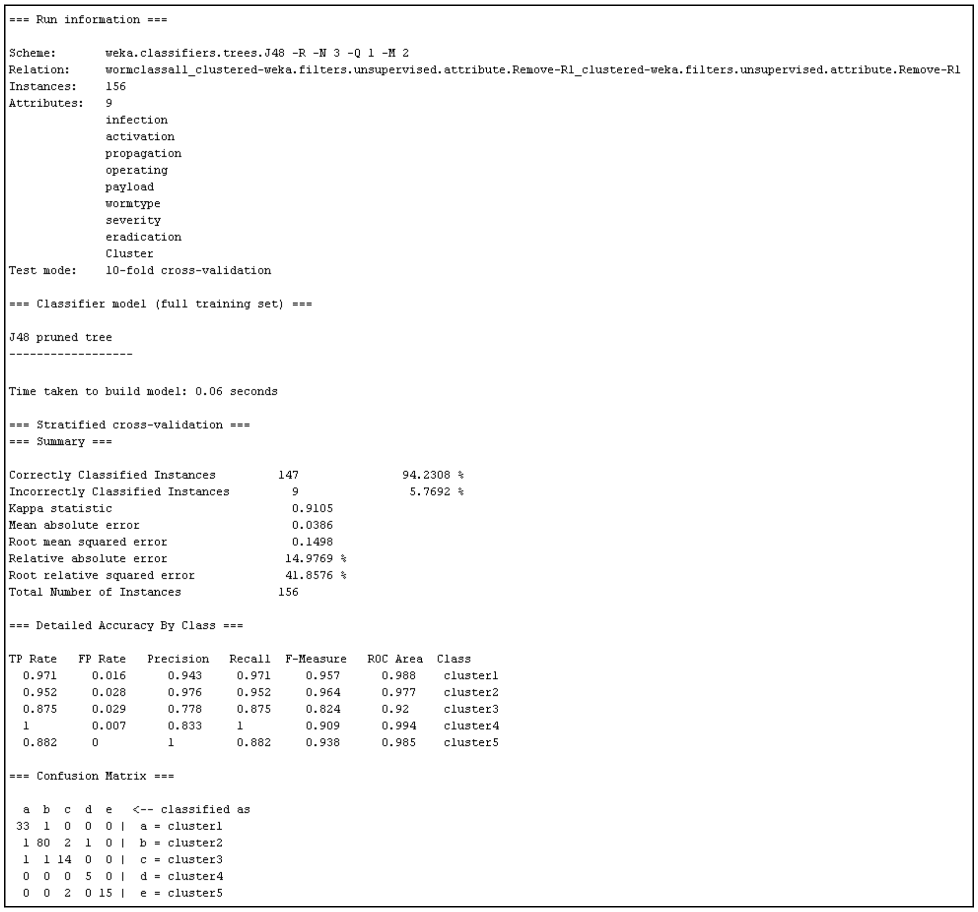
```
=== Run information ===

Scheme:       weka.classifiers.functions.SMO -C 1.0 -L 0.0010 -P 1.0E-12 -N 0 -V -1 -W 1 -K "weka.classifiers.functions.supportVector.PolyKernel
Relation:     wormclassall_clustered-weka.filters.unsupervised.attribute.Remove-R1_clustered-weka.filters.unsupervised.attribute.Remove-R1
Instances:    156
Attributes:   9
              infection
              activation
              propagation
              operating
              payload
              wormtype
              severity
              eradication
              Cluster
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

SMO

Time taken to build model: 0.8 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         151               96.7949 %
Incorrectly Classified Instances         5                3.2051 %
Kappa statistic                          0.9497
Mean absolute error                      0.2418
Root mean squared error                  0.3188
Relative absolute error                 93.6965 %
Root relative squared error             89.0741 %
Total Number of Instances              156

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
  0.971     0.008      0.971     0.971     0.971      0.99     cluster1
  0.976     0.042      0.965     0.976     0.97       0.97     cluster2
  0.938     0          1         0.938     0.968      0.996    cluster3
  1         0.007      0.833     1         0.909      0.997    cluster4
  0.941     0          1         0.941     0.97       0.991    cluster5

=== Confusion Matrix ===

  a  b  c  d  e   <-- classified as
 33  1  0  0  0 |  a = cluster1
  1 82  0  1  0 |  b = cluster2
  0  1 15  0  0 |  c = cluster3
  0  0  0  5  0 |  d = cluster4
  0  1  0  0 16 |  e = cluster5
```

III) Results for IBk Simulation



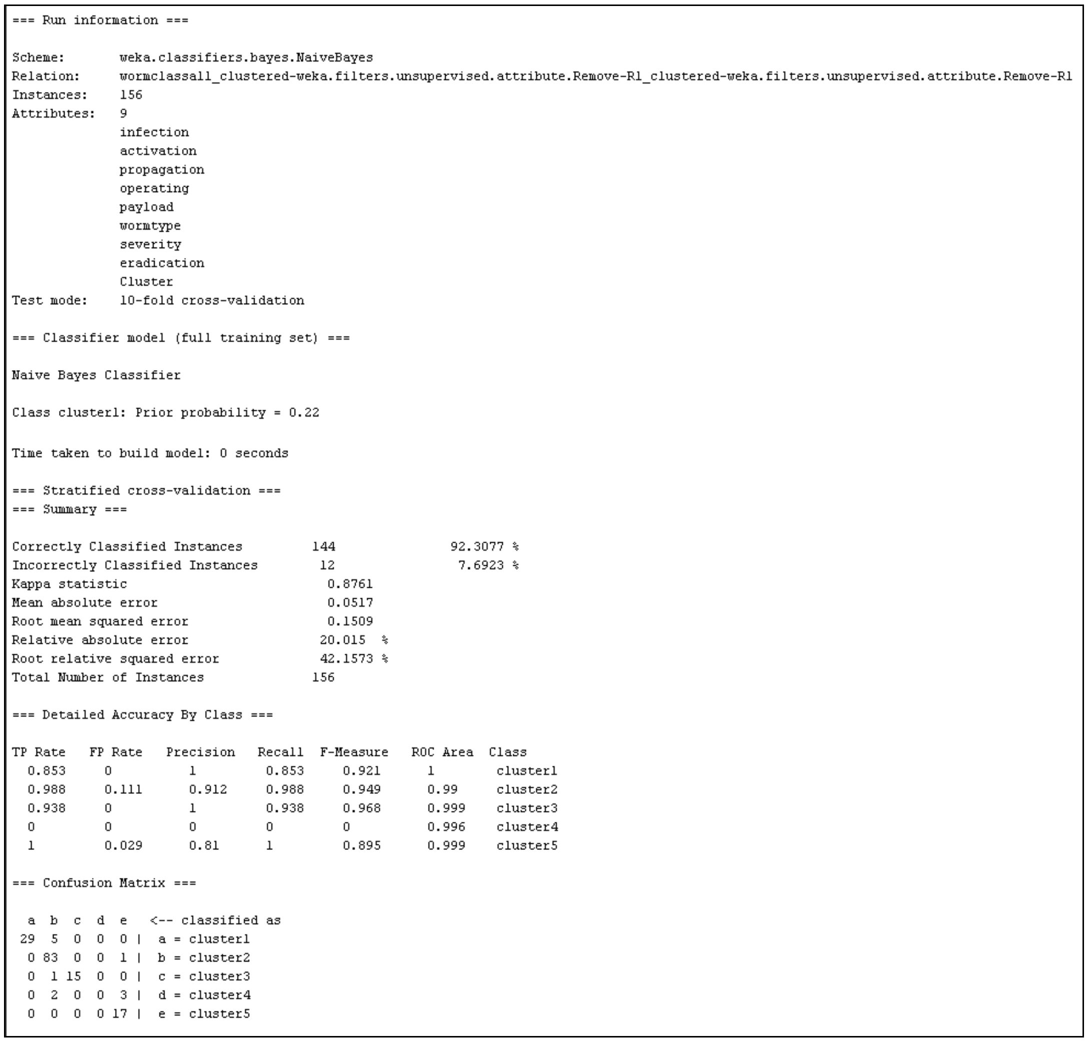
```
=== Run information ===

Scheme:       weka.classifiers.lazy.IBk -K 5 -W 0 -X -I -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""
Relation:     wormclassall_clustered-weka.filters.unsupervised.attribute.Remove-R1_clustered-weka.filters.unsupervised.attribute.Remove-R1
Instances:    156
Attributes:   9
              infection
              activation
              propagation
              operating
              payload
              wormtype
              severity
              eradication
              Cluster
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

IB1 instance-based classifier
using 3 inverse-distance-weighted nearest neighbour(s) for classification

Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         150               96.1538 %
Incorrectly Classified Instances         6                3.8462 %
Kappa statistic                          0.9389
Mean absolute error                      0.0299
Root mean squared error                  0.121
Relative absolute error                 11.5789 %
Root relative squared error             33.8016 %
Total Number of Instances              156

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
  0.941     0          1         0.941     0.97       0.993    cluster1
  0.988     0.069      0.943     0.988     0.965      0.988    cluster2
  0.938     0          1         0.938     0.968      1        cluster3
  0.8       0          1         0.8       0.889      0.988    cluster4
  0.941     0.007      0.941     0.941     0.941      0.997    cluster5

=== Confusion Matrix ===

  a  b  c  d  e   <-- classified as
 32  2  0  0  0 |  a = cluster1
  0 83  0  0  1 |  b = cluster2
  0  1 15  0  0 |  c = cluster3
  0  1  0  4  0 |  d = cluster4
  0  1  0  0 16 |  e = cluster5
```

IV) Results for J48 Simulation



```
=== Run information ===

Scheme:       weka.classifiers.trees.J48 -R -N 3 -Q 1 -M 2
Relation:     wormclassall_clustered-weka.filters.unsupervised.attribute.Remove-R1_clustered-weka.filters.unsupervised.attribute.Remove-R1
Instances:    156
Attributes:   9
              infection
              activation
              propagation
              operating
              payload
              wormtype
              severity
              eradication
              Cluster
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

J48 pruned tree
------------------


Time taken to build model: 0.06 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         147               94.2308 %
Incorrectly Classified Instances         9                5.7692 %
Kappa statistic                          0.9105
Mean absolute error                      0.0386
Root mean squared error                  0.1498
Relative absolute error                 14.9769 %
Root relative squared error             41.8576 %
Total Number of Instances              156

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
  0.971     0.016      0.943     0.971     0.957      0.988    cluster1
  0.952     0.028      0.976     0.952     0.964      0.977    cluster2
  0.875     0.029      0.778     0.875     0.824      0.92     cluster3
  1         0.007      0.833     1         0.909      0.994    cluster4
  0.882     0          1         0.882     0.938      0.985    cluster5

=== Confusion Matrix ===

  a  b  c  d  e   <-- classified as
 33  1  0  0  0 |  a = cluster1
  1 80  2  1  0 |  b = cluster2
  1  1 14  0  0 |  c = cluster3
  0  0  0  5  0 |  d = cluster4
  0  0  2  0 15 |  e = cluster5
```

V) Results for Naïve Bayes Simulation



```
=== Run information ===

Scheme:       weka.classifiers.bayes.NaiveBayes
Relation:     wormclassall_clustered-weka.filters.unsupervised.attribute.Remove-R1_clustered-weka.filters.unsupervised.attribute.Remove-R1
Instances:    156
Attributes:   9
              infection
              activation
              propagation
              operating
              payload
              wormtype
              severity
              eradication
              Cluster
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

Naive Bayes Classifier

Class cluster1: Prior probability = 0.22

Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         144               92.3077 %
Incorrectly Classified Instances        12                7.6923 %
Kappa statistic                          0.8761
Mean absolute error                      0.0517
Root mean squared error                  0.1509
Relative absolute error                 20.015  %
Root relative squared error             42.1573 %
Total Number of Instances              156

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
  0.853     0          1         0.853     0.921      1        cluster1
  0.988     0.111      0.912     0.988     0.949      0.99     cluster2
  0.938     0          1         0.938     0.968      0.999    cluster3
  0         0          0         0         0          0.996    cluster4
  1         0.029      0.81      1         0.895      0.999    cluster5

=== Confusion Matrix ===

  a  b  c  d  e   <-- classified as
 29  5  0  0  0 |  a = cluster1
  0 83  0  0  1 |  b = cluster2
  0  1 15  0  0 |  c = cluster3
  0  2  0  0  3 |  d = cluster4
  0  0  0  0 17 |  e = cluster5
```