

An Adaptive Approach to Detecting Behavioural Covert Channels in IPv6

Abdulrahman Salih

School of Science and Technology

A thesis submitted in partial fulfilment of the requirements of
Nottingham Trent University for the degree of

Doctor of Philosophy

May 2017

Copyright Statement

I confirm that this is my own work and the use of all materials from other sources has been properly and fully acknowledged. No part of this thesis has already been, or is being currently submitted for any such degree, diploma or other qualification.

Signature: _____

This thesis is copyright ©2017 by Abdulrahman Salih

Acknowledgements

Foremost, I would like to acknowledge that Allah's mercy was always with me before, through the achievement of this work and for ever. Secondly, I would like to express my sincere gratitude to my supervisor Dr. Xiaoqi Ma for the amazing honest support of my Ph.D study and research, for his immense knowledge, motivation, patience, and enthusiasm. His guidance helped me in all the time of research and writing of this thesis, more than 300 hundred emails and many phone calls during the project were dedicated by him to guide and support me.

As I would like to express my deeper respect to Prof. Dr Lars Nolle whom supervised me for the first year of my Ph.D study before moving back to Germany, his road map and intellectual guidance will never be forgotten. I also would like express my gratitude to my co-supervisor Dr Evtim Peytchev for his precious supports and advices I was honoured. Dedicated thanks to my colleagues and friends for their help and availability to share knowledge and new ideas in cyber security domain.

At last but not least, I would like to present my deepest gratitude and love to my beloved Mother for her prayers, anxiousness, care and continuous spiritual support to me during the preparation of this thesis.

Abstract

One of the most important techniques in data hiding is (Metaferography) covert channel, which recently has shown potential impacts on network and data security. Encryption can only protect communication from being decoded, meanwhile, covert channel is the art of hiding information in an overt communication as a carrier of information. Covert channels are normally used for transferring information stealthily. They are used to leak information across the network and to ex/infiltrate classified information from legitimate targets. These hidden channels violate network security and privacy polices, it is easy to embed but unlikely and almost impossible to be detected.

Despite of the obvious improvements in IPv6 components and functionality enhancements, there exist intrinsic security vulnerabilities. These vulnerabilities have ongoing implications on network security and traffic performance. Hence, they will create insecure environments in business and banking network, information security management and IT security. ICMPv6 is vital integral part in IPv6, as well as IPsec protocol, to mitigate and eliminate covert channels, the RFC standards and controls should be investigated intensively. Furthermore, incomplete implementation of IPv6 nowadays on all Operating Systems has not exposed the realm of this security protocol performance explicitly.

In this thesis, we present a novel Hybrid Heuristic Intelligent Algorithm coupled with enhanced Polynomial Naïve Bayes machine Learning algorithm. The framework is implemented in a supervised learning model to detect and classify covert channels in IPv6. The

proposed multi-threaded framework acts as an active security warden processing intelligent information gain and optimized decision trees technique to improve the security vulnerabilities in this new network generation protocol.

This new approach develops intelligent heuristic techniques for in depth packet inspection to analyse and examine the header fields of IPv6 protocol. Some of these fields are designated by the designer for quality of service (QoS), future performance diagnostic analysis, unfortunately, they are misused by "bad guys and black hats" to perform various network security attacks against vulnerable targets. These attacks cause immediate and ongoing damage to classified data. In order to prevent and mitigate these types of breaches and threat risks, a multi-security prevention model was created. Furthermore, advanced machine learning technique was implemented to detect, classify and document all current and future unknown anomaly attacks. The suggested HeuBNet6 classifier obtained highly significant results of 98% detection rate and showed better performance and accuracy with good True Positive Rate (TPR) and low False Positive Rate (FPR).

Keywords: covert channel, hybrid, heuristic, IPv6, ICMPv6, Multinomial, Naïve Bayes.

Publications

The fundamental parts of the work in this research have been presented and peer-reviewed by academics and professionals in International conferences, moreover they have been published as articles in electronic and hard copy format. The published papers are:

1. Salih, A., Ma, X. and Peytchev, E., 2015. Detection and classification of covert channels in IPv6 using enhanced machine learning. Proceeding in International Conference on Computer Technology and Information Systems (**ICCTIS'** 2015), Dubai, United Arab Emirates, 13-15 August 2015. <http://irep.ntu.ac.uk/27605/>
2. Salih, Abdulrahman., Xiaoqi Ma, and Evtim Peytchev.(2015) "New Intelligent Heuristic Algorithm to Mitigate Security Vulnerabilities in IPv6", International Journal for Information Security (**IJIS**),Volume 4, DOI : 04.IJIS.2015. http://irep.ntu.ac.uk/id/eprint/27592/1/PubSub5184_Ma.pdf
3. Salih, Abdulrahman., Xiaoqi Ma, and Evtim Peytchev.(2016). Implementation of Hybrid Artificial Intelligence Technique to Detect Covert Channels Attack in New Generation Internet Protocol IPv6, **Springer** Proceeding in Business and Economics, International Conference for Leadership, Innovation and Entrepreneurship as Driving Forces of the Global Economy (ICLIE),. 20-22 April 2016, Dubai UAE,. https://link.springer.com/chapter/10.1007%2F978-3-319-43434-6_15
4. Salih Abdulrahman, Zubair Adam and Jwaid Ali.(2017), "Detecting of Traffic Signature Vulnerability within DoS Attack in IPv6 Using Correlation Analysis",. Advances in Science,

Technology and Engineering Systems Journal, vol. 2, no. 5,
(2017). <http://astesj.com/v02/i05/>

5. Zubair Adam, Jwaid Ali and Salih, Abdulrahman.(2016),"Analysing Denial of Service Attack Traffic Signature in IPv6 Local Network Using Correlation Inspection",. proceeding TC 2016 - Future Technologies Conference 2016, San Francisco, USA, Technically Co-Sponsored by **IEEE**.
<http://ieeexplore.ieee.org/abstract/document/7821726/>

Contents

Copyright Statement	i
Acknowledgements	ii
Abstract	iii
Publications	v
Contents	vii
List of Figures	xii
List of Tables	xiv
Abbreviations	xvii
1 Introduction	1
1.1 Background and Motivation	1
1.2 Research Questions	4
1.3 Aim and Objectives	5
1.4 Research Methodology	6
1.5 Thesis Contributions	7
1.5.1 A New Structure for Security Detection Model	7
1.5.2 A Novel Intelligent Heuristic Algorithm	8
1.5.3 A New Primary Dataset for Covert Channel Attacks in IPv6	8
1.5.4 A New Shared Knowledge Framework	8
1.6 Thesis Organization	9

2	Literature Review	11
2.1	Information Hiding Techniques	12
2.2	Covert Channels in TCP/IP Protocol Suite	13
2.3	Related Work	15
2.4	Internet Protocol Version 6 (IPv6)	17
2.4.1	IPv6 Header Format	18
2.5	IPv6 Security Vulnerabilities	20
2.5.1	Extension Header Threats	21
2.5.2	Hop by Hop Extension Header Attack	23
2.5.3	Flow Label Threats	25
2.5.4	Routing Header Threat	26
2.5.5	ICMPv6 Threats and Multicast	26
2.5.6	Auto configuration and Neighbour Discovery Threats	28
2.5.7	Fragmentation Threats	29
2.5.8	IPsec Security Issues	30
2.5.9	Mobile IP Security Issues	30
2.6	Security Threats Posed by Covert Channel	30
2.6.1	IPv6 Covert Channels Characteristics	31
2.6.2	ICMPv6 Covert Channels Characteristics	35
2.7	The Need for New Detection Approach Against IPv6 Attacks	38
2.8	State of the Art of Machine Learning Application in Intrusion Detection Systems	39
2.9	Machine Learning Techniques	42
2.9.1	Naïve Bayes Algorithm (NB)	42
2.9.2	Multinomial Naïve Bayes (MNB)	44
2.9.3	Decision Trees C4.5 and Feature Selection	45
2.10	Summary	46
3	Proposed Detection Framework	48
3.1	Introduction	48
3.2	The Main Scenario	50
3.3	Probabilistic Approach	52
3.4	Data Aggregation Process	53

3.4.1	Data Collection Method	55
3.5	Suggested Model Stages	56
3.5.1	Raw Data Stage Capturing	56
3.5.2	Filtering and Data Analysis Stage	59
3.5.3	Data Pre-processing Stage	60
3.5.4	Data Training Stage	61
3.5.5	Detection and Classification Stage	62
3.5.5.1	Covert Channel Analyser	63
3.5.5.2	Multinomial Naïve Bayes Classifier (MNBC)	64
3.5.6	Issue Verdict Stage	64
3.5.7	Exporter and Validation Stage	65
3.6	Naïve Bayes Classifier	66
3.6.1	Information Gain and Gain Ratio	66
3.7	The Novel HeuBNet6 System Algorithms	69
3.7.1	Attack Simulation Process to Create Datasets	70
3.7.1.1	Create Training DataSet	73
3.7.1.2	Create Testing DataSet	73
3.7.2	HeuBNet6 Main	74
3.7.3	HeuBNet6 Running Modes	76
3.7.3.1	Data Training Mode	76
3.7.3.2	MNB Training Mode	78
3.7.3.3	Live Mode	80
3.7.4	Covert Channel Detection Engines	80
3.7.4.1	HeuBNet6 Intelligent Heuristic Engine (HIHE)	81
3.7.4.2	HeuBNet6 MNB Engine	102
3.7.4.3	C4.5 Feature Pruning	109
3.7.5	Anomaly Detection Reporting and Analysis	112
3.7.5.1	Alarm	112
3.7.5.2	Statistical Reporting and Analytics	113
3.7.6	Exporting Cross Validation ARFF Files	114
3.8	Summary	116

4	Experimental Methodology	117
4.1	Introduction	117
4.2	Network Topology	118
4.3	Setting Up Testing Environment	119
4.3.1	Virtual Machine Environment	119
4.3.2	Covert Channel Attack Tool	122
4.4	Attack Scenarios	123
4.4.1	Scenario One	124
4.4.2	Scenario Two	124
4.5	HeuBNet6 System Modules	124
4.5.1	Training DataSet Creation	124
4.5.2	Testing Dataset Creation	125
4.5.3	Testing Heuristic Intelligent Engine	126
4.5.3.1	Experimental Traffic Class Anomaly Covert Detection	126
4.5.3.2	Experimental Flow Label Anomaly Covert Detection	127
4.5.3.3	Experimental Hop Limit Anomaly Covert Detection	129
4.5.3.4	Experimental Next Header Anomaly Covert Detection	131
4.5.3.5	Experimental Payload Length Anomaly Covert Detection	132
4.5.3.6	Experimental Source Address Anomaly Covert Detection	135
4.5.3.7	Experimental Type-Code Anomaly Detection . . .	137
4.5.4	Computation of Class Label Using MNBC	140
4.5.5	Exporter	143
4.5.5.1	ARFF File	144
4.5.6	Alarm Flagger	145
4.6	Summary	147
5	Evaluation Methodology and Analysis	148
5.1	Cross Validation	149
5.2	Evaluation Metrics	151

5.3	Confusion Matrix	153
5.4	Precision Analysis	155
5.5	Evaluation of Covert Detection	158
5.5.1	Covert Detection Criteria	158
5.6	Critical Analysis	165
5.6.1	Detection Method	166
5.6.2	HeuBNet6 as a Rule Based System	167
5.6.3	HeuBNet6 as a Hybrid System	168
5.6.4	Statistical Report	168
5.6.5	Analytical Report	169
5.7	Weaknesses of Naïve Bayes Algorithm	170
5.8	Evaluation Results	171
5.9	Summary	171
6	Conclusions and Future Work	173
6.1	Conclusions	173
6.2	Summary of the Contributions	177
6.2.1	A New Structure for Security Detection Model	178
6.2.2	A Novel Intelligent Heuristic Algorithm	178
6.2.3	A New Primary Dataset for Covert Channel Attacks in IPv6	178
6.2.4	A New Shared Knowledge Framework	179
6.3	Limitations of The Project	179
6.3.1	Ethical Consideration	180
6.4	Future Work	180
	Appendices	182
A.1	HeuBNet6 and ICMPv6 with TypeCode Algorithm	182
A.2	Investigated Request For Comments Protocols	187
A.3	Flow Label Manager "Corpus"	188
A.4	Sample of Training Dataset	190
	Bibliography	193

List of Figures

2.1	TCP/IP Protocol Suite Architecture [1]	15
2.2	The header format of IPv6 [2]	19
2.3	The IPv6 128 bit address calculation	20
2.4	TCP/IPv6 Extension Headers' types and format [1]	22
2.5	ICMPv6 Header Format [3]	27
2.6	IPv6 header field values in pcap data	32
2.7	Covert Channels in Routing Extension Header [4]	34
2.8	Identified storage Covert Channels in the Hop-by-Hop Options Extension Headers [5]	34
2.9	Integral ICMPv6 packet header format in IPv6 [2]	36
2.10	The encoded ICMPv6 message format in JPCAP	36
2.11	The decoded ICMPv6 message format	37
2.12	Machine Learning Techniques Classification [6]	43
3.1	The prisoners' problem with communication monitored by warden [7]	51
3.2	Proposed Covert Channel Behavioural Detection Model	57
3.3	Linux Netfilter traffic netflow	58
3.4	Filtering Data Type from Captured Packets	60
3.5	Example of Header Field Conversion to NB-Class Instances	61
3.6	Nominal File List for Training MNB Model	79
3.7	Flow Label Corpus	88
3.8	Feature Nominal-NB-Class Map	102
3.9	Feature Nominal-NB-Class Instances	102
3.10	Flow Label Corpus	112

4.1	Suggested LAN Topology	122
4.2	Suggested topology for training dataset creating	125
4.3	Traffic Class anomaly detection topology	127
4.4	Flow label anomaly detection topology	129
4.5	Hop Limit anomaly detection topology	130
4.6	Next Header anomaly detection topology	133
4.7	Payload Length anomaly detection topology	134
4.8	Source Address anomaly detection topology	136
4.9	Type_Code anomaly detection topology	138
4.10	Parameter Problem anomaly detection topology	139
4.11	ARFF file format created for evaluation process	145
4.12	Alarm notification of an anomaly detection from HeuBNet6	146
5.1	Estimation accuracy with holdout method [8]	150
5.2	Overall HeuBNet6 accuracy in comparison to other classifiers	152
5.3	HeuBNet6 Confusion Matrix	155
5.4	Performance of precision Level of MNBC	156
5.5	A sample of NB classified output data format	158
5.6	Echo Request evaluation graph	160
5.7	Echo request evaluation logarithmic graph	160
5.8	ICMPv6 Header Covert Detection graph	162
5.9	ICMPv6 Next Header evaluation logarithmic graph	163
5.10	Echo Request Anomaly versus Next Header Anomaly Evaluation logarithmic graph	164
5.11	ICMPv6 Next Header evaluation logarithmic graph	165

List of Tables

2.1	Technical Comparison between IPv4 and IPv6 [9]	20
2.2	IPv6 Option Headers [2]	24
2.3	Identified Covert Channels in IPV6 Header Fields [5]	32
2.4	Possible Covert Channels in Routing Header [10]	34
2.5	Format of Covert Channel in the Hop-by-Hop Options Extension Headers [11]	35
2.6	ICMPv6 Type network permitted messages [2]	37
2.7	ICMPv6 Network dropped messages [2]	37
3.1	ARFF Header format for Validation Process	54
3.2	ARFF Header format for Validation Process	61
3.3	Feature Nominal-NB-Class Map	62
3.4	Nominal values to Naïve Bayes Format	63
3.5	ARFF Header format for Validation process	66
3.6	Data format in ARFF output file prior to detection process	66
3.7	Identified Covert Channels in IPV6 Header Fields [5]	66
3.8	Original Nominal Values and NB formats for Selected Fields with Example Values	71
3.9	Selected Header Fields with Their Characteristics Values	72
3.10	Feature Groups and Combination Estimations	77
4.1	Firewall static wall	118
4.2	Testbed devices IPv6 link-local/global addresses with MAC	120
4.3	HeuBNet6 experiment configuration check list	121
4.4	Traffic Class Corpus	126

LIST OF TABLES

4.5	Detection of covert traffic class result using NIHA and MNBC . . .	127
4.6	Covert detection of flow label using NIHA and MNBC	129
4.7	Covert detection of Hop Limit anomaly using NIHA and MNBC . .	131
4.8	Covert detection of Next Header anomaly using NIHA and MNBC .	133
4.9	Covert detection of Payload Length anomaly using NIHA and MNBC	135
4.10	Covert detection of Fake Source Address anomaly using NIHA and MNBC	136
4.11	Covert detection of Type _Code using NIHA and MNBC	139
4.12	Detection of Parameter problem covert data using NIHA and MNBC	140
4.13	Training Data in Nominal Values to Naïve Bayes Format	140
4.14	Results of new attacks detection using NIHA & MNBC	143
4.15	Ranking of the attributes using C4.5 with InfoGain	144
5.1	Overall HeuBNet6 Accuracy Detection	151
5.2	Sample training dataset used in Evaluation Process	151
5.3	MNBC Data Size Used in Confusion Matrix for covert anomaly detection	154
5.4	Confusion Matrix Showing the Actual Predicted Class	154
5.5	Values of Confusion Matrix with the Actual Predicted Class	155
5.6	Sensitivity and Specificity of HeuBNet6 performing 10-Fold Cross Validation	156
5.7	Precision of HeuBNet6 Model Accuracy in Comparison to Current Classifiers	157
1	Corpus of Flow Label Manager	189
2	HeuBNet6 Training Dataset Sample	191
3	HeuBNet6 Training Dataset Sample	192

List of Algorithms

1	Create_Training_DataSet	73
2	Create Testing DataSet	73
3	HeuBNet6_Main	75
4	HeuBNet6_Data_Training_Mode	76
5	HeuBNet6_MNB_Training_Mode	79
6	HeuBNet6_Live_Mode	80
7	HeuBNet6_Intelligent_Heuristic_Algorithm	81
8	HeuBNet6_MNBC	107
9	HeuBNet6_C45_Feature_Pruning	110
10	HeuBNet6_Alarm	113
11	HeuBNet6_Export_ARRF_Files	114
12	HeuBNet6_ICMPv6_TypeCode_Anomaly_Detection_Algorithm	182

Abbreviations

ARRF	Attribute-Relation File Format
EM	Experimental Messages
ETN	Extension Type Numbers
HIHE	HeuBNet6 Intelligent Heuristic Engine
IANA	Internet Assigned Numbers Authority
MLA	Machine Learning Algorithm
MLD	Multicast Listener Discovery
MNAD	Managed Network Address Directory
MNBC	Multinomial Naïve Bayes Classifier
ND	Neighbor Discovery
NIHA	Network Intelligent Heuristic Algorithm
PPM	IPv6 Parameter Problem Message
PTBM	Packet Too Big Messages
RBS	Rule Based System
RFC	Request For Comments
TCP	Transmission Control Protocol
TEM	Time Exceeded Messages
TEM	Time Expired Messages
THC	The Hacker's Choice ToolKit
UDP	User Datagram Protocol
UIM	Unallocated Information Messages

Chapter 1

Introduction

1.1 Background and Motivation

Any wired or wireless communication channel should be subject to security policies that comply with most international restriction rules and acts. These rules apply on all types of information transmitted by these channels which may contain sensitive data; however, it is vital for the data to be refrained from unauthorized access. Given that fact, these transferred messages are again subject to compromise through some sort of powerful penetration "*hacking*" communication techniques which operate stealthily, known as covert channel. We can categorize these communication channels into two main types: *overt communication channel* and *covert communication channel*. Overt channels are official and basically acknowledged, and comply with all network security policies and privacy acts. Meanwhile covert communication channels unintentionally exist in such a way that any type of information would be transmitted and accessed in unrestricted way and unnoticed by network wardens, violating standard system security policies and privacies.

The United States Department of Defence in 1985 presented a definition for covert channels [12] as "*A communication channel that allows a process to transfer information in a manner that violates the system's security policy*". Lampson [13] also defined covert channels as a monolithic system works in such a mechanism that leaks information from a high security level to low security level which is meant to

be disallowed to access the systems if this mechanism was not used. Essentially, these channels have not been explored or designed by researchers to be used for transferring information purpose. Hence, Murdoch and Lewis [14] claimed that covert channels can leak information intentionally by using a single channel which basically has not been created for the purpose of violating access control policy. On the other hand, side channel unintentionally can perform the same function in leaking information. Internet Protocol version 6 (IPv6) is a new generation and the successor of the networking protocol version 4. It enables Windows, Linux and any other operating system users to communicate with other users over the internet [15]. IPv6 was introduced in 1998 by the Internet Engineering Task Force (IETF) in order to replace IPv4, technically based on the standard specification on Request for Comments RFC 2460 draft project [16].

Covert channel modifies existing overt traffic protocols' formats and structure targeting either entirely the protocol's fields, or any vulnerable components of the protocol that may not be obvious for warden or network administrators. The security impacts of IPv6 storage covert channel are not less effective than an immediate security threat such as Denial of Service (DoS) or Distributed Denial of Service (DDoS). Furthermore, the damage that this type of technique causes in a short period of time through controlling compromised PC's on legitimate secure network upgrades the threat level to a higher security scheme. In this context, an attacker may steal classified information over a long time which will cause multiple unwanted implications of security breach [17].

An obvious example of using covert channel as a security threat causing catastrophic results is the significant attacks performed by covert codes hidden in a sensitive hardware that was purchased by a victim [18]. The codes were activated to disrupt the functioning of the hardware to perform an attack against Syrian surveillance radar over-watching the Israeli's bombing raid in 2007. This type of achievement depends on the effective stealthy method and its activation mechanism to make covert channel attack a good implementation choice [17].

The taxonomy of covert channel is a challenge due to the complexity of the protocol design and techniques used to tackle these new security vulnerabilities. Packet headers might not contain sufficient information to allow such detection methodology. Researchers have found that implementing some machine learning

algorithms merely such as Naive Bayes would offer a limited detection rate reaching 71% [19, 20]. However, using an additional sophisticated algorithm which combines supervised Machine Learning techniques such as Naive Bayes or Multinomial Naive Bayes and heuristic analysis to classify and detect covert channels in IPv6 is an innovation for this problem domain. Heuristics are mental short cut or "rule of thumbs" that give some guidance on how to do a tasks, but it does not guarantee solutions consistently [21]. Heuristics may be used to determine the specific rules for solution generation. The algorithm is coined in this research as an Intelligent Heuristic Algorithm which analyses the filtered headers, creates the attack instances and label them with the class type, finally makes the decision whether each packet contains covert or not. None of the previous researchers attempted to detect almost all types of covert channels in IPv6 header; rather, the attempts followed the general pattern of detection techniques implemented in traditional firewalls, IDS and IPS. Furthermore, it was hard to find models and frameworks that specifically implemented Machine Learning methodology such as Naive Bayes or Multinomial Naive Bayes to detect covert channels in IPv6 which offers a distinctively accurate detection rate. This fact is according to the latest research on hybrid AI techniques used in IPv6 covert channel detection [21].

Additionally, Lucena *et al* [5] suggested the detection of covert channels using active warden in various complex scenarios which resulted in large consumption of time and resources [7] as a consequence of not employing machine learning techniques. Therefore, the need for more research to implement Machine Learning techniques in this problem domain is considerable. This sort of approach carries a clear multi-layer security model which will add a shared knowledge framework as a different way to tackle such sophisticated covert channels in IPv6.

Despite the fact that overt and covert channels are important to be considered in computer security analysis, this project aimed at designing a hybrid framework to detect only covert channel behaviour. In overt channels the data transmission is performed and complied with the existence of security policies, whereas in covert channels the communication path allows an unauthorized process to transfer information in such a way that violates the system security policy and privacy. Storage covert channels characteristics in

IPv6 and ICMPv6 header fields were investigated.

1.2 Research Questions

To survive with the ever-increasing network security threats utilizing the inherited Internet Protocol vulnerability in IPv6 by embedding storage covert channel, innovative detection and elimination techniques are vital to stop security attacks such as embedded metaferography techniques (covert channel) against this protocol. The primary hypotheses and questions are:

- **Hypotheses**

1. Using information hiding techniques has been a driving force behind privacy and network security policy violation in IPv6.
2. Attackers can use different data hiding techniques apart from steganography in IPv6 to infiltrate classified data or ex-filtrate sensitive information from legitimate targets.
3. Intruders can communicate using IPv6 vulnerabilities to embed data and send it to their targets. Covert channels are used to perform network attacks against vulnerable targets.

- **Research Questions**

1. What is the feasibility of a new prevention model to detect and eliminate covert channel attacks in IPv6?
2. What effective countermeasure methods can be used to mitigate security implications and the associated risks in IPv6?

After examination of the above questions, the process led to potential sub questions which may enrich the approach to using different techniques to mitigate such security threats on IPv6. The sub questions are:

- (a) Is it possible to develop a classifier to categorize an arbitrarily sized database of labelled attack instances and test the conditional independence status of the attributes?

- (b) Can a supervised Machine Learning technique be developed to create a knowledge based framework in order to detect and classify covert channels in IPv6?
- (c) Is it possible to optimize the accuracy of covert channels classification using advanced feature selection technique?

A new approach to detecting these hidden communication channels in IPv6 header fields will be proposed. This approach consists of Intelligent Heuristic Algorithm coupled by an advanced Multinomial Naïve Bayes Classifier using enhanced field selection techniques.

1.3 Aim and Objectives

This research aims to create a new knowledge-based framework to analyse IPv6 security vulnerabilities, and to detect behavioural hidden communication channels within this network layer. This framework will mitigate and eliminate the security threats against IPv6. It will create primary data which represents covert channel characteristics of IPv6. The empirical process will be performed through enhancing advanced Machine Learning techniques to solve such problem in network security. Suggested Multinomial Naïve Bayesian classifier (MNBC) is adding a new direction of thinking to create network security systems towards mitigation of future complex security problems and unknown attacks.

The attack hypothesis behind the project is that intruders can use different data hiding methods to communicate over open data channels [22]. TCP/IP covert channel is one of the techniques mimicking different attacks to ex-filtrate/infiltrate sensitive data from machines, routers or classified data storage servers. This action violates security policies and the three essential Information security principles: **Confidentiality**, **Integrity** and **Availability** (CIA) of any targeted organization with no regards to any privacy legislations [23, 24].

The objectives of this research are as follows:

1. Explore the trade-off between channel type, size and the ease of detection by investigating the performance of selected covert channels and their countermeasures.

2. Propose a new and different countermeasure to detect storage hidden channels that are used to ex/infiltrate classified information from legitimate targets.
3. Suggest and develop a Machine Learning Algorithm such as Multinomial Naive Bayes to improve the detection probability process of covert channels in IPv6.

1.4 Research Methodology

In order to achieve the aim and objectives mentioned in Section 1.3 for this research, a clear understanding is needed of the problem domain and the research questions clarifying the scope of the project. Furthermore the following methodology and investigations will be conducted:

1. Study and understand the IPv6 protocol architecture and investigate the security implications of TCP/IPv6 suite protocols.
2. Study and examine the IPv6 security vulnerabilities to identify IPv6 covert channels and their characteristics via related RFC's standards.
3. Analyse different security attacks and threats caused by covert channels against legitimate targets.
4. Design an essential network topology and security tool to simulate the hypothesised network attacks. Using qualitative method to create primary data for experimental examination.
5. Investigate current different detection and prevention approaches in IPv4 and IPv6, then analyse their security implications.
6. Investigate Machine Learning Algorithms in network Security.
7. Investigate different feature selection techniques such as C45, Info Gain to differentiate, categorize and customize the optimal value of the pre-processed data based on testing and performance results.

8. Investigate the performance of MLA accuracy and false rate used in the suggested security model.
9. Design and implement a reliable and effective security framework to detect and classify covert channels in IPv6 protocol.
10. Evaluate and validate the suggested detection Model via using a parallel software with different mechanism and test plan.

1.5 Thesis Contributions

The contributions of the proposed framework in this thesis are the guidelines for analysing, classifying and detecting IPv6 protocol security vulnerabilities. Due to the lack of IPv6 covert channels benchmark data, primary attack instances will be created through simulation of local attacks tool on a separated LAN topology.

As a clarification for the primary contribution of this research, to the best of our knowledge, this is the first effort that suggests a multi-threaded security system for IPv6 to mitigate such complicated security threats created by covert channels attacks. Furthermore, to the best of our knowledge, in a part of the attempts performed [5, 17, 25, 26, 27], there is no sufficient data (known and unknown attack) of using covert channels in IPv6 similar to what was achieved in this thesis. This contribution elaborates the novel covert channel's model that implements Intelligent Heuristic Algorithm creating the necessary primary data. This data consists of nearly 6 millions multi instances attributes which represent various types of covert channel attacks and can be in future research.

The major contributions presented in this thesis are provided in Sections [1.5.1-1.5.4](#).

1.5.1 A New Structure for Security Detection Model

A multi-threaded security structure is designed for IPv6 to detect covert channels (information carrier) attacks and threats. The flexibility of this structure is offering a new attempt to deal with high speed traffic flow and throughput. This part will be seen in Section [3.5.2](#).

The model uses advanced data mining techniques as a supervised machine learning algorithm in an enhanced mode of Multinomial Naive Bayes classifier (MNB) coupled with a novel heuristic analysis approach. This model is an adaptive security approach which can be used as a modern paradigm for threat detection and classification in future accumulative research. This part of contribution will be seen in Section 3.6.

1.5.2 A Novel Intelligent Heuristic Algorithm

The research contributes a novel Network Intelligent Heuristic Algorithm (NIHA) as an essential security system model to analyse, assess and create covert channel characteristics in IPv6 and ICMPv6 packets. This will optimise the weight and value of each attack instance. This part of the contribution will be seen in Section 3.7.

This algorithm is the core element of the security detection model as it provides the means to specify and manage detection policies and specifications that are used to detect covert channels in IPv6.

1.5.3 A New Primary Dataset for Covert Channel Attacks in IPv6

The creation of new primary data through the use of NIHA is a leap into ground breaking areas of IPv6 covert channels. New attack instances are created as an outcome of the empirical research progress. The primary data for the initial development of the project are potentially impressive to the testing and evaluation phases. Furthermore, this data can be used for future research and similar models evaluation particularly in TCP/IPv6 security problem domain. This part will be seen in Section 3.7.

1.5.4 A New Shared Knowledge Framework

A new shared knowledge framework is presented and implemented in this research supported by the developed security model and a new sample research dataset produced by NIHA. The knowledge base can be redeployed at all critical points of

an organisation's network topology. It is envisaged that further research emanating from this work will produce the amalgamation of newly discovered attacks due to changes IPv6 specifications acquired through the wider research community.

1.6 Thesis Organization

The organization of thesis will be as follows:

Chapter Two illustrates an extensive study about the security vulnerabilities in TCP/IPv4 and IPv6. An in-depth investigation and examination of the security protocol structure, Request for Comments (RFCs) and Internet Assigned Numbers Authority (IANA) rules are performed in relation to the information security and covert channels that cause serious violations in privacy and network security policy. The main characteristics of the covert channels in IPv4 and IPv6 are discussed. The potential threats caused by IPv6 and ICMPv6 headers which are the core of the issue will be discussed. Additionally, the different approaches proposed by academic researchers and professionals to solve this particular problem in network security are discussed. Finally the need for such detection approach is discussed and justified.

Chapter Three presents the proposed architecture of the covert channel detection and classification system. The main modules of the framework are elaborated, including packet filterer and data analyser, covert channels analyser and detector, characteristic classifier, verdict issuer and exporter. The proposed network intelligent algorithm is explained theoretically and technically. The technical details behind the new suggested algorithm are explained intensively. The enhanced Multinomial Naive Bayes classification is presented and discussed along with the feature selection methodology.

Chapter Four demonstrates the empirical steps in the system design and implementation. Furthermore, the experimental software development platform and environment are explained and discussed. The creation of primary data which is a vital part for the project with regards to the current cutting edge technology is elaborated. Behavioural covert channels are tested and examined

thoroughly using the differentiation of targeted parameters and attributes in ICMPv6 and IPv6 header fields. The tools used to create and simulate attack instances are discussed. The results produced are highlighted. Validation and evaluation of the framework were performed using Weka system.

Chapter Five discusses the results, outcomes and analysis from the testing phases of the framework. An evaluation of the results is performed through the assessment and testing plan analysis. Association analysis is used to evaluate the framework performance. The overall critical analysis, drawbacks and limitations of the project are highlighted and discussed. The distinctive parts of the project are discussed in comparison to current and relevant projects in network security.

Chapter Six summarises the major contributions of this research work. Furthermore, the potential impacts on the IPv6 security vulnerabilities are highlighted. A brief explanation of the achieved aims and objectives of the project is presented. Future work and planned stages for the project are discussed. Finally, potential future directions of the relevant areas of the project are highlighted.

Chapter 2

Literature Review

In Sections 1.1, 1.2 and 1.3, a brief overview has been given on covert channels, their security implications, and the research questions have been discussed. Network covert channel has a mechanism that could be used to leak data across network protocols. Many studies have described covert channels in network protocols such as TCP/IP version 4 and ICMPv4 systematically, but not IPv6. This chapter discusses some other network protocols covert channels, then describes some elements and properties used in this research. Finally, it presents an in-depth clarification of relevant covert channels in IPv6.

This chapter is organised as follows: Section 2.1 describes the information hiding techniques for embedding covert channel. Section 2.2 discusses TCP/IP covert channels generally. Section 2.3 discusses the related work of covert channels detection. Section 2.4 explains some technical specifications of IPv6 and its potential modified fields. Section 2.5 explains a list of IPv6 security vulnerabilities. Section 2.6 discusses the specific security threats posed by covert channels. Section 2.7 argues about the need for new covert channel detection approach in IPv6. Section 2.8 presents the state of the art of using Machine Learning in Intrusion Detection Systems. Section 2.9 explains Machine Learning technique types used in anomaly detection. Finally, Section 2.10 is the summary. In Chapter 3 more addressed covert channels in IPv6 and its implementation will be discussed in details, although this chapter comes prior to the suggested security approach to tackle covert channels in IPv6. It is important to

understand the methodology used in the project.

2.1 Information Hiding Techniques

Information Hiding was presented by Parnas [28] stating that the critical design of any system should not be disclosed to end users who benefit from the system and no underlying details of an internal program functions should be revealed. An example of information hiding is a calculator which hides the calculation process from the user. There is a flexibility in information hiding for a programmer to modify a program in order to gain the opportunity of secretly inserting source code into modules. This flexibility makes the access and development easier in the future for the programmer. This is one of the many advantages in information hiding.

There are three types of techniques [29] that information hiding can be used: cryptography, steganography and covert channels (metaferography). Wendzel [30] agreed with Petitcolas to call covert channels formally as "metaferography" which is a part of information hiding discipline and should be harmonized with the naming convention for cryptography and steganography. The basic idea was presented by Pfitzmann [31] when informally he was engaged in a workshop meeting in 1996 to demonstrate that information hiding was a concept of embedding `<datatype>` into an embedded `<datatype>` called covert data in which a steganographic key could be a part of this embedding process despite of whatever data type was hidden into the nested or embedded `<datatype>`. The three types of information hiding techniques [31] are as follows:

1. **Cryptography:** This term came from an ancient Greek word *κρυπτο* (kryptos) which means "hidden" writing, as any type of disorganised text messages would be indecipherable without using a secret key. Plaintext was transformed into cipher text using a key phrase [32]. The writing was there but it was unreadable. This type of technique was used during wartime when the American Civil War by the Southern Confederacy implemented a confederate Cipher Disc to protect battle communication details from unauthorised access by the enemy [31].

2. **Steganography:** This term is also derived from a Greek word *στεγανον* (steganos), which means "covered" writing, particularly hiding a message within another message. The existence of the message is hidden so it could be called "security through obscurity" in order to protect the message from unauthorised access [29, 31]. This type of technique was also used during the American Revolutionary War against the invasion of British army. There are some types of stego implementation using content rich media, such as photo, image, audio or video, to conceal a message.
3. **Metaferography (Covert Channel):** This term is originated from the Greek word *μεταφερο* (metaferos), which means "carried" writing. Metaferography is used to hide the actual message, meanwhile steganography hides the message. Metaferography conceals the message within the carrier or transferring mechanism. Covert Channel is a vital part among metaferography techniques, relating to modern data communication electronically [24].

Metaferography was used by ancient Greek to protect messages by writing them on surfaces of wooden tablets then covering them with wax in order to make them look like normal wax tablets [29]. According to Savacool [24] metaferography brings consistency to the nomenclature and can be used alongside with other types of information hiding such as cryptography and steganography. Metaferography could describe the pitch (extend) of domain concerning covert channels while a covert channel describes the actual implementation of information hiding.

Due to repetitive work of cryptographic and steganographic techniques in TCT/IP [5, 33, 11], the problem domain of this research needs to be narrowed down. Eventually, it will focus on detection of information carrier techniques (metaferography) in IPv6 only.

2.2 Covert Channels in TCP/IP Protocol Suite

The diversity of network protocol fields and their implementation has given a clear opportunity to embed covert information. Jankowski *et al.* [34] presented a

method using Ethernet, ARP and TCP to create covert data. A similar method performed by David *et al.* [35] used IP and Ethernet frames to insert encoded covert data into the payload header. The essential technical aspects of TCP/IP protocol suite is described in RFC793 [36] as shown in Figure 2.1. There are some extensions of the TCP [37] which has additional header options as other researchers [15, 23, 38, 39] claimed that these extension headers have been identified to hold stenographic coding. TCP/IP headers can serve as carrier of stenographic covert channels according to its RFC's presuming the possibility of taking one set of values as these values could be detected by passive warden [7, 20, 40, 41]. TCP does not provide a reliable channel, however as a connection-oriented protocol it preserves its reliability properties if a network shows packet loss, recording and duplication [42]. This particular aspect gives TCP an advanced reliability in implementation and flow control which offered scope to stenographic coding. Meanwhile, IP protocol itself has no capabilities to provide any type of assured stream reliability [14, 43].

A covert channel is used to ex-filtrate or infiltrate confidential data of a higher level system to a lower level system or vice versa and can be used in steganography. It deals with network issues which are not explicitly applied to multi-level security (MLS). One of the characteristics of network covert channel is that it enables the transferring of data without drawing any attention of being detected, thus violating network security policies [14].

In system security analysis process, some vital steps should be done in regards to evaluation of both overt and covert communication channels. Covert channels can be categorized into two main groups: storage covert channels and timing covert channels [12, 44]. A storage covert channel is used to transfer data through altering the storage attribute of an object such as a value of a field or a file's contents in a specific directory in IPv4 or IPv6. Timing covert channel functions by altering timing attributes (i.e. the timing interval of a network packet) or the sequence of an event (i.e. network packet sorting time) [45].

Other types of covert channel exist which are neither timing nor storage types called behavioural covert channels as they depend on the receiver and sender's behaviours. Furthermore, Anthony and Sabelfeld [35, 44] argue that other types of covert channels based on probability distributions, resource exhaustion, and

power consumption are also available. Some researchers [20, 44, 46] think that covert channels are dual-purposed as they could create potential security threats such as Trojan horses or botnets.

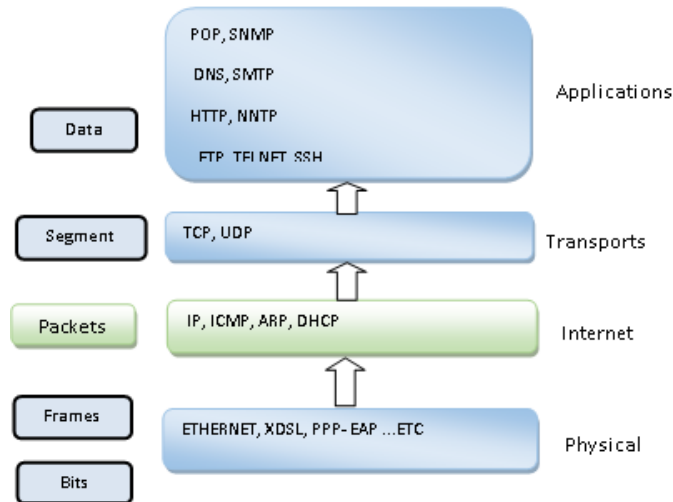


Figure 2.1: TCP/IP Protocol Suite Architecture [1]

2.3 Related Work

Previous researchers [23, 39, 47] in network covert channels focused on IPv4 while fewer researchers concerned about security vulnerabilities of the new generation protocol IPv6 [5, 17, 48, 49, 50] due to its incomplete implementation. Furthermore, most of the investigated covert channels were the storage type rather than the timing channels [42, 51, 52, 53, 54].

A few tools exist for setting up network covert channels using different protocols such as TCP, HTTP and ICMP [55, 56, 57]. The section packet of the protocol can hold large portions of data which makes it easy to hide information and facilitate the transfer due to the unorganised structure if compared to the headers. These headers are designed for protocol improvement in the future, therefore it is easy to encode covert channels in the unused bits or the reserved part of the packet frame. Lack of intensive values in the protocol standards and RFC's drives Intrusion

Detection Systems (IDS) and Firewalls to dismiss or grant exception to these headers [58] during inspection.

Handel and Sandford [59] detected that using the unused field of type of service (TOS) header or the flag field can create covert channel in TCP/IP header [60]. Ahsan *et al.* [45] used Flag field header "Don't Fragment" (DF) bit as a covert channel. He proposed five methods to use TCP, IGMP and ICMPv4, as one of these approaches meant to sort data in IPsec protocol. The DF bit can be set to arbitrary values if the attacker knows the Maximum Transfer Unit (MTU) size in case fewer packets of the MTU size are sent.

Hintz [61] suggested to manipulate TCP's urgent pointer to send hidden data deceiving the system to obtain high priority data in case the URG bit is not set. Trabelsi [62] and Lucena *et al.* [5] proposed numerous covert channels in the new generation protocol header fields such as traffic class, flow label, hop by hop and other fields.

Rowland [52] proposed a covert channel technique through multiplying each byte of the hidden data in TCP/IPv4 protocol suite header fields by number 256 to be used instantly as IP ID. There is a major requirement by RFC 0791 for IP standard which considers IP ID to unambiguously identify IP packet for a short time interval [63]. Furthermore, Rutkowska [64] suggested an advanced hidden channel using ISN of TCP in Linux by encrypting the data in ISN fields giving an identical random. Dunigan [65] presented the same example as Rowland did implement the covert channels in the header fields of the OSI network model [17, 5].

Murdoch and Lewis [14] criticised previously proposed hidden channel technique using ISN, expecting to get a different outcome of implementation than any OS gives. This is done through tailoring the developed covert channels for Linux and Open BSD in order to make the ISN covert channels distribution more real and normal. Qu *et al.* [66] used Time to Live and hop limit fields to embed hidden information so as Lucena [5] but unfortunately neither scheme took into account that initial TTL values will be chosen by the sender whereas normal TTL changes in network [58].

Zander *et al.* [33] suggested converted covert channel after analysing techniques proposed by Qu and Lucena. This technique was more difficult to detect. Dyatlov, Castro, Kwecka and Van Horenbeeck proposed a variety of techniques to embed

covert channels in HTTP protocol headers [56, 67, 68]. Sohn *et al.* [12] suggested passive warden technique using Support Vector Machine (SVM) to detect Ahsan's and Rutkowska's covert channels.

However, this approach is not desirable for unambiguous features as he proposed steganography in ISN. Additionally, SVM is unable to detect complex and embedded aspect without additional advanced technique to understand TCP's mutuality [55]. ICMPv4 tunnelling technique also has been suggested by Loki [46] to embed covert data into the payload field holding ICMP_echo request and echo_reply packets. The receiver unwraps the transmitted packets to execute the commands and send them back to repeat the process with ICMP packets.

2.4 Internet Protocol Version 6 (IPv6)

Vital changes in IPv6 need to be addressed. These modifications have been designed and provided in such a way that will enrich our investigation and protocol security analysis. As mentioned above, IPv4 has some well-known limitations which do not comply with the current growth of the internet: address space depletion, large routing tables and lack of security. Consequently, IETF has decided to develop IPv6 in order to eliminate some of these limitations [61], focusing on quality performance, ease of configuration as well as solving network management issues. The IPv6 core specifications have been defined in different Request of Comments (RFC's), mostly in RFC 2460 [9, 2]. IPv6 provides a range of improvement over IPv4 such as simplicity, big address space, easier auto-configuration, simple routing header, flow labelling capability, and enhanced security through a compulsory use of IPsec protocol.

The changes and removed fields in IPv4 indicate some major key differences between IPv4 and IPv6 [53, 58, 69]. During the transformation of the Internet in the 1990's from a research network to a commercialised network, concerns were raised about the ability of IPv4 to accommodate emerging demand. In 1993 the Internet Engineering Task Forces (IETF) began the design and standardisation process to develop the next generation of Internet Protocol that would address among other issues the predicted exhaustion of available IPv4 addresses. A new

standard collection resulted out of this research is called IPv6. This was developed over several years as many aspects of this solution would be evolved within IETF. Finally, in 1998 a new version of IPv6 was built [16, 49].

The functionalities and components of IPv6 architecture will be discussed next, but a brief comparison between IPv4 and IPv6 headers can provide some answers. The changed and removed components of IPv4 are explained as follows [2, 70, 71, 72]:

1. The header length field is eliminated in IPv6 because the length is fixed in IPv6.
2. The service type field is eliminated in IPv6. The traffic class and flow label fields together take over the function of the service type field.
3. The total length field is eliminated in IPv6 and replaced by the payload length field.
4. The identification, flag, and offset fields are eliminated from the base header in IPv6; instead they are included in fragmentation extension header.
5. The Time to Live (TTL) field is called hop limit in IPv6.
6. The protocol field is replaced by the next header field.
7. The header checksum is eliminated because the checksum is provided by upper layer protocols.
8. The option fields in IPv4 are implemented as extension headers in IPv6.

2.4.1 IPv6 Header Format

The IPv6 header format as shown in Figure 2.2 reflects the address size and the ultimate number of offered IP addresses is shown in Figure 2.3. The features which IPv6 protocol brings to plate are described in several RFC's. The packet headers fields in IPv6 are similar to IPv4 except some removed and changed field. The header has no other header extensions as carried out in frames and consists of 8 fields, as detailed below [1, 16]:

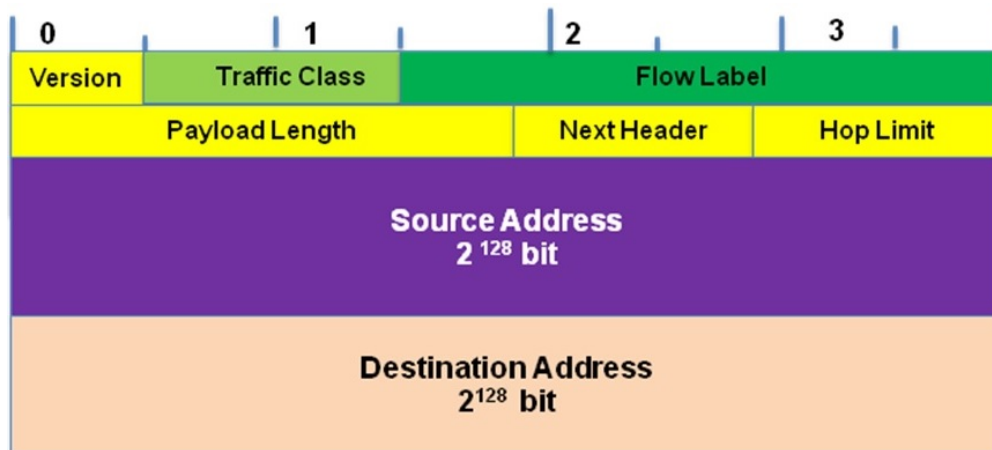


Figure 2.2: The header format of IPv6 [2]

- **Version:** consist of 4 bits (0.5 byte). It identifies the protocol version which has the value of 6 in binary system.
- **Traffic Class:** consists of 8 bits (1 byte), used by the sources and routers to identify the packets belonging to the same traffic class, so it distinguishes between packets and different classes based on priorities.
- **Flow Label:** consists of 20 bits (2.5 bytes) and is used as a label of data flow.
- **Payload Length:** consists of 16 bits (2 bytes) and indicates the length of packet data field which is a balance of IPv6 packet following header.
- **Next header:** consists of 8 bits (1 byte) and indicates the extension header immediately following the IPv6 packet header.
- **Hop Limit:** consists of 8 bits (1 byte), and is decremented by one by each node which forwards the packet. This is the maximum number of hops that an IPv6 packet can perform, similar to IPv4's Time to Live (TTL). When the hop limit reaches zero, the packet is discarded.
- **Source Address:** consists of 128 bits (16 bytes) and indicates the original source of the packet.

- **Destination Address:** consists of 128 bits (16 bytes) and indicates the original destination of the packet.

IPv6 is 128-bits means:
 $2^{128} = 340,282,366,920,938,463,463,374,607,431,768,211,456$
 $2^{128} = 2^{32} * 2^{96}$ then the IPv6 addresses are more than IPv4 as 2^{96}
 $= 79,228,162,514,264,337,593,543,950,336$ times the number of IPv4 addresses so it's 79
trillion trillion addresses.

Figure 2.3: The IPv6 128 bit address calculation

Table 2.1 shows the most distinguished comparison between IPv4 and IPv6.

Table 2.1: Technical Comparison between IPv4 and IPv6 [9]

Feature	IPv4	IPv6
Address field space	23 bit	128 bit
Address Configuration	Static DHCP	Auto, DHCPv6, Static
Address for each Interface	Single	Unlimited, Link Local Address
Subnetting	Inconsistent	Node's IDs have 264 bits
Extension Headers	No	Yes
Broadcast	Yes	No
Fragmentation	Routers & Hosts	Only Hosts
ICMP	Optional	Mandatory
Link Layer	ARP	ND for ICMPv6
NAT	Mandatory	Not Applicable
IPsec	Optional	Mandatory

2.5 IPv6 Security Vulnerabilities

To create an interoperable protocol the Internet Engineering Task Force (IETF) presented IPv6 specification's details that should be followed by implementers [73]. This fact has changed the security equation in dealing with the protocol

ambiguities in some areas. Additionally, some unforeseen security issues, which have not been considered when the protocol was designed, contribute in these new vulnerabilities [5, 74]. Hackers and security researchers can easily explore the reasonable hints between specifications and practical deployment. An important issue should be highlighted that IETF sometimes performs technical revisions of protocols, but somehow for specific protocol types, IETF offers the IP system developers to correct protocol deficiencies [73].

There are some vulnerabilities within the IPv6 packet header fields, thus security issues arise once new network software and hardware implementation tolerate manipulation of these vulnerabilities. Therefore, a network administrator must be aware and understand the security issues in IPv6 and should know how to detect them. The security threat issues of IPv4 are outside the scope of this project and will not be discussed.

There are three main types of vulnerabilities in IPv6 according to RFC 4942 classification [75, 73]: the inherited vulnerabilities in the design, the transition vulnerabilities, and the deployment vulnerabilities. This research will focus on some of the inherited security vulnerabilities in the header. There are some other issues [16, 76] which can not be fixed unless the scratch design of IPv6 itself is changed. In this section, the major security issues are discussed.

2.5.1 Extension Header Threats

Unlike IPv4 which uses Options field [62], IPv6 as shown in Figure 2.4 uses extension headers. This is either to present the packet information related to transport layer of TCP and UDP or to maximize the protocol functionality. Next Header (NH) field has the ability to identify the extension headers within an IPv6 header. Specifically, 8 bits are the carrier of NH data starting from bit 48 within the IPv6 header [9]. Extension header is used to indicate the next header in an IPv6 packet. Furthermore, this header is designed to be placed after destination address field, and before the upper layer, which is considered as a part of IPv6 payload. Despite the fact that a packet can have more than one extension header, these headers are not required for processing in routers except hop-by-hop option header. These headers as shown in Figure 2.4 are usually

presented in the following order [9]:

- Hop-by-Hop options headers.
- Routing headers.
- Fragment header.
- Destination option header.
- Authentication Header (AH).
- Encapsulation Security Payload (ESP) header.

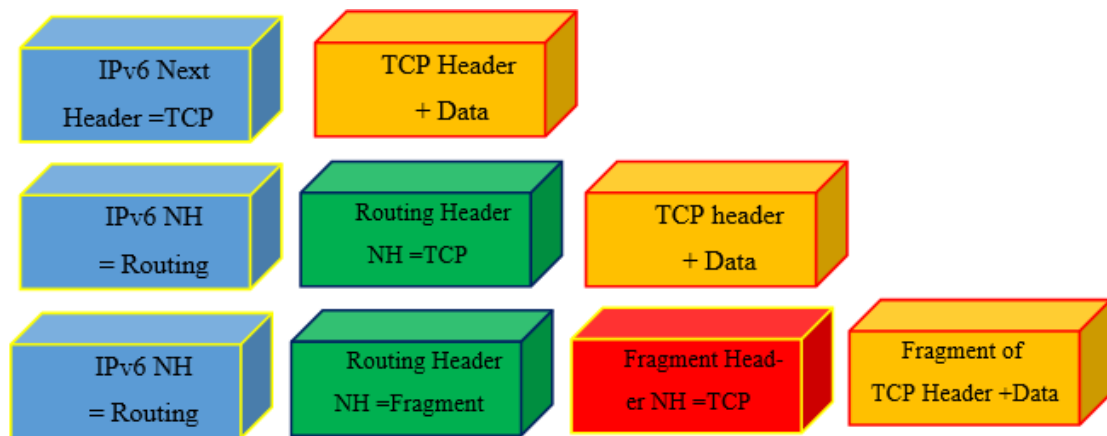


Figure 2.4: TCP/IPv6 Extension Headers' types and format [1]

These headers [77, 78] support IPv6 protocol and provide some advantages such as security, Jumbo-gram and mobility. The security architecture of this Internet protocol is explained in RFC 2460. The last two headers AH and ESP also have been described in RFC's 2401, 2402 and 2406 to ensure that both headers could be used in two different modes: transport mode and tunnel mode. In end-to-end connection all existed payload in IP packets require encryption or authentication in transport and tunnel mode. Techniques such as authentication and encryption exist between two gateways like firewall using IPsec to mount the packet data (the IP header and the payload) with a "wrapper" IP packet [40, 79].

Initial analysis of the protocol was performed on potential covert channels in IPv6 beside the Internet Assigned Number Authority (IANA) rules and standards. In this analysis, specified covert channels have been identified due to lack of specific limitations. However, the mandatory IPsec protocol in IPv6 created an obvious impact on the discovered covert channels. Each extension header has a unique identification number ID which enables it to determine the next header's value. This ID [2, 9] contains the header's type that informs the receiver to parse the header to follow as shown in Table 2.2. These numbers have been set up by IANA and synchronised with IPv4 protocol numbers [2]. Despite each type of these header's own vulnerability, the most distinctive vulnerability in the extension headers is that an attacker can craft an IPv6 packet with extension header manipulation to create a Denial of Service attack [5, 9, 62].

2.5.2 Hop by Hop Extension Header Attack

Hop by hop header is one of the six extension headers in IPv6 header specified in RFC 2460 [80]. It is the only header that its packets are processed by routers along from source to destination and should be checked by every node through the packet journey. The flexibility of its defined and undefined option types, as well as its variable length [9, 49] offers an opportunity that covert channels can be created. Using the inconsistency of the options to deliver information through modifying the packet fields creates this type of covert channel. Attackers can misuse this extension header option when it contains more than one option and is set to various sizes. Eventually, different types of DoS attacks can be performed. Routers are responsible to check the options in the header as it is unlikely easy to control [5, 81]. This attack could degrade the routers' or node's CPU performance. Furthermore, padding options Pad1 and PadN are used to ensure the standard 8 octet boundaries size. In this case, altering the default value of the padding option with a non-zero value creates covert channels communication [5, 26].

This extension header should be the first one in header sequential extensions; misusing this extension will create a security issue in packet data transmission. All of these security issues have been investigated and analysed by researchers [5, 9, 17] and some detection mechanisms have been proposed. However, no final

Table 2.2: IPv6 Option Headers [2]

Next Header ID No	Specs, RFC's and Function
0	Hop-by-Hop Options, raising a flag to the router, RFC2460, <ul style="list-style-type: none"> • Each packet has a single type, • It must be the first header used, • Its estimate by all nodes and hops upon delivery path, • Capacity to support maximum Jumbo payload >65.535 and <4 billion bytes, • Either Jumbo or Fragment should be used alone
6	Transmission Control Protocol (TCP), RFC 2460, <ul style="list-style-type: none"> • It provides a reliable packet deliver
17	User Datagram Protocol (UDP), RFC 2460, <ul style="list-style-type: none"> • It provides unreliable packet delivery
43	Routing Options, RFC 2460, <ul style="list-style-type: none"> • Tracks the future visited nodes during the packet transmission, • Dismissed by some nodes on the way to destination
44	Fragmentation Option, FRC 2460, <ul style="list-style-type: none"> • The Source node can only fragment a packet no bigger than the Maximum Transmit Unit, It is processed by Destination node
50	Encapsulation Security Payload (ESP), RFC2406, <ul style="list-style-type: none"> • It provides Confidentiality, • It provides Encryption security technique, • It has a connectionless integrity, • offers Authentication of original data
51	Authentication Header (AH), RFC 2402, <ul style="list-style-type: none"> • It offers Connectionless Integrity, • It offers authentication of original data
58	ICMPv6, RFC 2463, <ul style="list-style-type: none"> • It's an integral part of IPv6 and must be fully implemented by every node in IPv6 network, • It present error encountered messages while processing packet transmission, • Facilitate Neighbour Discovery
59	No next header availability, RFC2460, <ul style="list-style-type: none"> • This is the last header so there is no possibility to create or send further more headers
60	Destination Options, RFC2460, <ul style="list-style-type: none"> • It provides optional data transmission to destination, • It is tested by the destination node only.
88	Enhanced IGRP (EIGRP) version 6 (EIGRPv6), IPv6 for cisco routing protocol
89	Open Shortest Path Version 3(OSPFv3), RFC 5340, IPv6 for OSPF
103	PIM-SIM, RFC 4601, Protocol Independent Multicast Space Mode
115	L2TPv3, RFC3931, Layer2 Tunnelling Protocol version 3
135	Mobile IPv6, RFC3775, Mobility header

resolution has been found to this particular problem. Krishnan *et al.* [82, 83] suggested to deprecate this extension header from the protocol or stop it from creating new option definition which means that IPv6 should skip this header's processing. Zander *et al.* [26] suggested to limit the bandwidth rate of the packet with hop-by-hop extension header. Unfortunately, his suggestion will lead to drop the packet whenever the CPU signalled very high [20].

2.5.3 Flow Label Threats

As shown in Figure 2.2, IPv6 header consists of 40 bytes. Compared with IPv4 header, seven header fields have been removed and five fields have been changed. A new field was added called flow label which obviously has some good functionalities such as indication to Quality of Service (QoS) and also can be used as a load balancing signal by inserting a pseudo-random value to protect against spoofing attacks [78, 84]. This field's specification was altered continuously despite the argument mentioned in RFC 6294 and some other updated specifications in two standards RFC 3697 and RFC 6437. The IPv6 group [49, 58] has received some proposals to deal with such issue. Flow label has three major vulnerabilities:

- The flow label exists only theoretically but is hardly used. This field is ignored or passed unchanged in packet forwarding and it is hardly used in practice in IPv6 implementation. However, once its value is set to Zero [77, 78, 85] it causes two types of DoS attacks: forging large number of IPv6 packets holding various values, and recording the forged packet header. Consequently, this will give a failure to accept legitimate host for not including the same extension header like the counterfeited packet.
- Possible covert channels implementation through zero value set of flow label. This can be done by setting a false value which will impose the intermediate device to perform wrong services, assuming that it has a default behaviour such as not modifying flow label value [23, 86].
- Flow label is neither protected nor included in transport pseudo-header checksums, thus, when malicious code changes it, it is difficult to be detected [5, 78]. Obviously, information leakage could be performed if the

flow label value is predicted by the attacker, hence altering the value will engage the router to deny services [74, 78, 87].

More details about this type of covert channel will be explained and discussed in the suggested algorithm in Section 3.7.4.

2.5.4 Routing Header Threat

IPv6 has an optional extension header and there are six types specified in RFC 2460. Using these headers introduces some security vulnerabilities: routing header is one of these headers with a value of 43 as shown in Table 2.2. This value is used to list one or more intermediate nodes which will be visited on its way to destination [5, 7]. This header has two types of routing headers: type 0 (RH0) for source routing indication and type 2 for mobile IPv6. Multiple addresses of the intermediate nodes could be found in a single RH0. As a result of this, each destination of any packet will be replaced passing any network layer hop which processes the router header [20]. Possibly, a packet with intermediate node will be dealt with as a source and it is considered as a security vulnerability.

On the other hand, when packet filtering can not process routing headers, an attacker can easily do one of the following: generate malicious packet with routing header containing a victim address as covert channel, publicly access addresses bypassing firewall which basically does not check an existing routing header extension, build a packet with a multiple processing possibility between two RH0 within the packet, and use the amplification ability of the packet between two remote routers [50]. This will apparently lead to network traffic congestion which is caused by a legitimate packet [3, 20]. Abley *et al.* [88] discussed that no legitimate packet will be transferred in this way and it is unlikely to prove the possibility of the exploitation of RH0 in IPv6 packet transmission.

2.5.5 ICMPv6 Threats and Multicast

Internet Control Message Protocol (ICMP) in IPv4 could be blocked mostly to improve security and eliminate threats. It is an optional component within IPv4

and all ICMP messages should be filtered. However, in IPv6, ICMPv6 as shown in Figure 2.5 is an integral part and it can not be blocked [2, 87]. Routers and hosts depend on this protocol; it performs the fault-isolation function in order to handle various error messages such as destination unreachable, packet too big, time exceeded, source quench, redirect and parameter problems [9, 89]. Another set of message exchange techniques used in ICMPv6 by routers and hosts are called Multicast Listener Discovery (MLD) and Neighbour Discovery (ND) Protocol. MLD mechanism enables routers to learn about other multicast addresses on the same link, which ultimately leads to security threats and enable attackers to easily gain access into the addresses. This action can be performed by sending forge packets to routers, then in response the router will send back a list of addresses [90, 91].

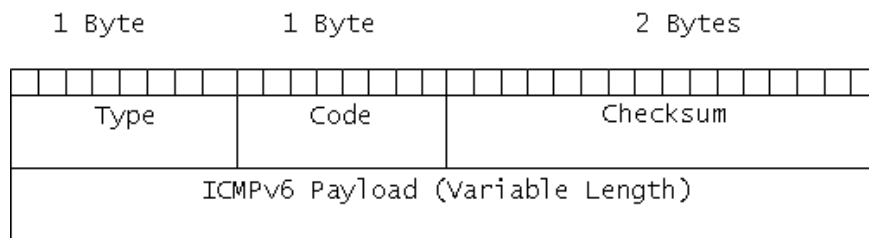


Figure 2.5: ICMPv6 Header Format [3]

Despite the fact that ICMPv6 must be fully implemented by every node according to RFC 4443 [5, 92], it can not do anything if the protocol itself commits an error. It reports error encountered in processing packets in addition to other inter-layer functions such as diagnostics. It produces two types of messages: error notification and information notification messages using two main elements in the protocol: Type and Code fields to distinguish between services [9, 26, 81]. Possible security attacks are likely to occur through these two fields such as Denial of Service (DoS), Man-in-the-Middle (MITM), and spoofing attacks [26, 49, 93]. Each of these messages carries a next header value of 58 as shown in Table 2.2. The Type value for message specification in a range of 1-127 is set for error messages, and 128-255 for notification messages.

The arbitrary content of the ICMPv6 payload may carry different type of data according to these messages and the operating system types. However, sometimes ICMPv6 packets carry insignificant or null values that indicate existence of potential covert channel [57, 71, 84].

2.5.6 Auto configuration and Neighbour Discovery Threats

Neighbour Discovery Protocol is based on ICMPv6 type messages [94, 95, 96] and provides a number of functions that are discussed later:

- Type 133, Router Solicitation (RS).
- Type 134, Router Advertisement (RA).
- Type 135, Neighbour Solicitation (NS).
- Type 136, Neighbour Advertisement (NA).
- Type 137, Redirect message.

IPv6 has two types of auto-configuration: stateful and stateless, stateless auto-configuration has several mechanisms to obtain an IPv6 address: First, generating an interface ID of IPv6 address from the 48 bits of the node's MAC address by using EUI-64 mechanism; second, using privacy extension address mechanism by generating the interface ID randomly; third, generating the interface ID using Cryptographically Generated Address (CGA) method [40, 85]. Stateful auto-configuration is used through DHCPv6, which should either support or provide general information (i.e. gateway DNS server, etc.), then only addresses should be auto-configured or DHCP should provide all details including address information [20, 57]. Neighbour Discovery Protocol (NDP) is an important part in IPv6 which is used to discover any neighbour in LAN network. Any live node in IPv6 network can perform this function. RFC 3756 [96] has specified some threats and vulnerabilities on NDP [96]. This protocol [63, 97] has a vital status within IPv6 functionalities and deals with three types of DoS threats [5, 20, 57].

The first type of DoS threats are:

- Non router/routing threats.
- Exploited messages including NS Neighbour Solicitation/NA Neighbour Advertisement spoofing.
- Neighbour Unreachability Detection Failure (NUDF) and Duplicate Address Detection (DAD).

The second type of DoS threats are:

- Router and routing in such form of malicious to last hop router.
- Default router is 'down or killed'; good router is considered as a 'bad' router.
- Redirect message spoofing, bogus on link prefix and parameter spoofing.

The third type of DoS threats are:

- The remotely exploitable attacks which include reply attack and NDP DoS attacks.

2.5.7 Fragmentation Threats

In IPv6 the source node is responsible of packet fragmentation and not the destination node. The Minimum Transfer Unit (MTU) should not be less than 1280 octets; thus, the intermediate nodes can not handle this in IPv4 [20, 51]. RFC 2460 rules [92] state that any packet less than 1280 octets will be discarded. Attackers can misuse this chance to achieve the DoS attack against the victims hosts [34]. This type of threat can be performed by exploiting datagrams break down to overbear the target networks, or transmitting fake UDP or ICMP packets [49].

2.5.8 IPsec Security Issues

The enhancement of IPsec in IPv6 is seen in providing three functions: Authentication Header (AH), Encapsulation Security Payload (ESP), and the Internet Key Exchange (IKE). This enhancement is used as a plug-in the IPsec framework [84]. This protocol is standard, strong and has extensive mechanisms through providing security for upper layers. It combines three elements to achieve this target: secret key protect service, security associate and mobile secret key management [12, 98]. IPsec is not able to provide secure support between applications, particularly in enterprise networks [98]. Security issues arise with this protocol because it does not solve all security issues such as DoS and Distributed Denial of Service (DDoS) through password and secret key attacks. In transport and tunnelling mode, a secret key exchange is needed, where both sender and receiver are vulnerable due to unawareness of what has been transmitted [95, 98].

2.5.9 Mobile IP Security Issues

Mobility is complicated due to complex design of IPv6. There is a security concern about the normal operations in mobile IPv6 such as authentication and authorization of the mobile host in a foreign network [95, 98]. The option header is used in mobility to store the 'original' address of a mobile host, meanwhile it uses the mobile address in the IPv6 header, which eventually allows spoofing attack. This is because providing false information to the legitimate target on the home agent diverts legitimate traffic. This drawback occurs when mobility is not used by default in normal networking, therefore there is no previous solution to mitigate this [95].

2.6 Security Threats Posed by Covert Channel

The primarily objectives of covert communication channels are summarised below:

- Theft of proprietary information in a stealthy way. This will lead to violate security policies of an organization or government through leaking sensitive

and intellectual property. This occurs to existing portable and non-portable storage media in or out of an organizational computer network. It will result in an increase in hidden network transmission activity and cause serious harm [37, 99].

- Delivery of malicious executable program code through the mechanism of installed covert channel with a system level privileges on a legitimate target. This will cause DoS against wide range of systems [5, 84].
- Signalling/Control mechanism for executable program code (Botnet). This type of program consists of distributed network computers with malicious codes established on all members of Botnet. Once activation time is due a signal or a control channel mechanism will activate the remote system which called "robot". Examples include Tribunal Flood Network (TFN) and Loki [100, 101].

The impact of data loss or misuse of network resources is significant through covert channels and other malware or physical theft. The loss of corporate secrets and client personal information, as well as other types of saved corporate data can be devastating to an organization financially and legally, especially when client trust is lost [53].

2.6.1 IPv6 Covert Channels Characteristics

The protocol dimension and fields values in network simulation language or PCAP data according to RFC 2460 [9] are shown in Figure 2.6. Covert channels exist in each filed through modifying its value. These channels can be classified to two taxonomy types: variable and predictable, according to the specifications given in RFCs [41, 78, 84]. A detectable hidden channel flags an existing variation. Meanwhile, if there is a variable channel, signal is given for indicating a limited modification [26]. Covert channels' characteristic in IPv6 are explained in Table 2.3.

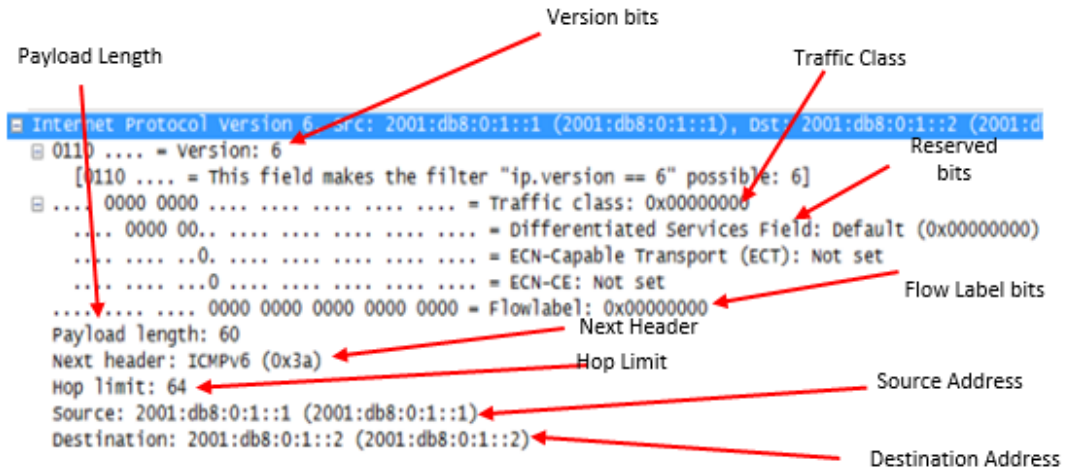


Figure 2.6: IPv6 header field values in pcap data

Table 2.3: Identified Covert Channels in IPV6 Header Fields [5]

ID	Field	Covert Channel	Bandwidth
1	Traffic Class	Set a false Traffic Class	8 bits/packet
2	Flow Label	Set a false Flow Label	20 bits/packet
3	Paylaod Length	Increase value to insert extra data	Various
4	Next Header	Set a valid value to add an extra extension header	Various
5	Hop limit	Increase/decrease value	~1 bit/packet
6	Source Address	Set a false address	16 bits/packet
7	ICMPV6_Type	Set false Type value	8 bits/packet
8	UCMPv6_Code	Set false Code value	8 bits/packet

Figure 2.6 is a screen shot of IPv6 PCAP data captured by Wireshark application. The headers are explained as follows:

1. **Traffic Class:** A false set traffic class field value [49] as the bandwidth is 8 bits per packet indicates that the field is modified and used as covert channel. This allows an intermediate node to change values when forwarding packets. Reusing this field as covert channel will create noise as the attacker should be aware about this issue. An error correction mechanism must be created to reduce the available bandwidth [84].
2. **Flow Label:** This field can be fabricated through sending 20 bits of data

per packet by the attacker. The authentic flow labels are pseudo-randomly and uniformly selected numbers, ranging from 1 to $0XFFFFFF$. The receiver should preserve the same condition once this fake flow label is created [5, 7].

3. **Payload Length:** Extra data can be appended in the end of payload packet and its value can be increased. Its bandwidth depends on the original size of the packet, but the modified packet cannot be larger than 65,535 bytes. The Maximum Transfer Unit (MTU) should not be exceeded when traffic is modified, because the intermediate nodes of IPv6 do not fragment packets. If exceeded, the packet will be dropped [5, 77]. If encryption is used in payload packets without authentication, stego-techniques [25] are suitable to be used. If authentication is used, extra steps will be needed by the attacker to maintain the covertness of the channel because the payload length will be included in the AH Integrity Check Value (ICV).
4. **Extension Header:** These extension headers are not examined or processed by the intermediate nodes in a communication path. An attacker can change the next header content to insert a whole extension header covertly. Accordingly, the payload needs to be increased by the attacker.
5. **Hop Limit:** Setting an initial hop limit value "h" can manipulate the hop limit value of sub sequence packets. By checking the variation of the hop limit values of packet traversing the attacker, covert message location can be interpreted.
6. **Source Address:** An attacker can forge the source address field to send 16 bytes of hidden data. However, this type of misusing the field will be detected quickly due to the existing mechanism of spoofing detection. Consequently, packets will be discarded.
7. **Hop-by-Hop Option Header:** Every transmitted packet needs to be checked by every node. Hop-by-hop options header carries optional information such as option types, defined and undefined, and its variable length. Furthermore, the extension header offers the opportunity for high bandwidth covert channels [17, 25].

Next Header (1 byte)	Header Extension Length (1 byte)	Routing Type 0 (1 byte)	Segment left (1 byte)
Reserved (4 bytes)			
Addresses (16 bytes)			

Figure 2.7: Covert Channels in Routing Extension Header [4]

8. **Routing Header:** Any packet while transmitting should pass by a list of intermediate nodes heading to its destination. There are two types of routing headers: Type 0 (RH0) as shown in Figure 2.7, it is used for source routing indication and type 2 as shown in Table 2.4 is used for mobile IPv6. An attacker can easily generate malicious packet with routing header containing a victim address as covert channel [95].

Table 2.4: Possible Covert Channels in Routing Header [10]

ID	Field	Covert Channel	Bandwidth
1	Routing Type: 0 Reserved	Hidden Data	4 bytes/packet
2	Routing Type: 0	Set one or more false addresses	~2048 bytes/packet

The protocol specifications [84] clarify that option type field is an octet structure. This type has three sub-fields: the first two bits specify which action should be taken once an unrecognised option is received, the next bit determines the possibility of the option data which can be changed and can hold a covert attack with the Jumbo-gram size between 0-65,535 bytes, and the last five bits represent the option number when the entire octet is used.

Ext Header (1 Byte)	Header Extension Length (1 Byte)	Option Type (1 Byte)	Option Data Length (1 Byte)	Option Data (Variable Length or specified in the Option Data)
------------------------	---	----------------------------	--------------------------------------	---

Figure 2.8: Identified storage Covert Channels in the Hop-by-Hop Options Extension Headers [5]

Table 2.5: Format of Covert Channel in the Hop-by-Hop Options Extension Headers [11]

ID	Field	Covert Channel	Bandwidth
a	Option Type: Jumbogram	Insert or create a jumbogram	Various
b	Option Type: Router Alert	Set a false router alert	2 bytes/packet
c	Option Type: PadN	Set a false padding value	Up to 256 bytes/packet
d	Option Type: Unknown	Fabricate one or more options	Up to 2038 bytes/packets

2.6.2 ICMPv6 Covert Channels Characteristics

In this section, covert channel as security vulnerability in ICMPv6 will be analysed. ICMPv6 as shown in Figure 2.9 is a vital component and an integral part of IPv6 communication process and must be fully implemented by every IPv6 node. The command ping6 was used to see the captured data in various formats. A dissected format of integral ICMPv6 in IPv6 is shown in Figure 2.9. The encoded format of these messages from a performed packet inspection is shown in Figure 2.10. The decoded PCAP data format of the same message is shown in Figure 2.11.

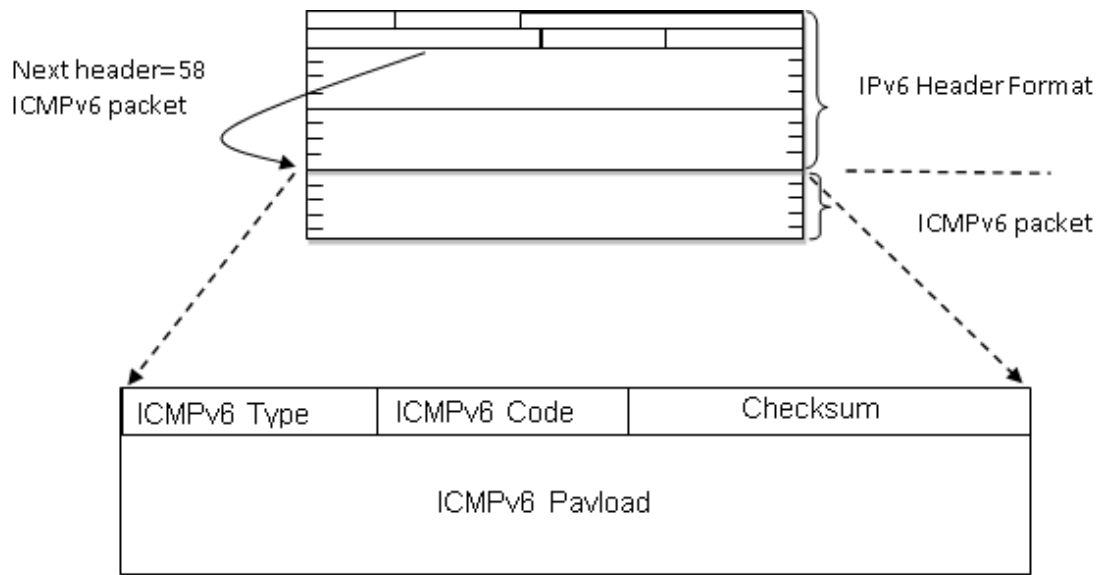


Figure 2.9: Integral ICMPv6 packet header format in IPv6 [2]

ICMPv6 creates two types of messages: Information Notification and Error Notification using fields Type and Code to differentiate services, which can be manipulate for Denial of Service (DoS) and spoofing attacks [2, 90].

```

00 50 56 c0 00 08 00 0c 29 b4 27 7b 08 00 45 00  .PV.... ).' {..E.
00 1c 00 00 40 00 40 01 7f 0e c0 a8 9d 80 c0 a8  ....@.@. ....
9d 01 08 00 e8 eb 0f 13 00 01  ....

```

Figure 2.10: The encoded ICMPv6 message format in JPCAP

Figure 2.11 shows the ICMPv6 header message format. These messages have a next header value of 58 which includes the following Type values for message specification: the range 1-127 for error messages and 128-255 for information messages. The Internet Assigned Number Authority (IANA) list of ICMPv6 type numbers gives more details [102, 103].

An ICMPv6 message contains: Type field as shown in Figure 2.11 with one byte and the Code is one byte as well, and the Checksum field designated (2 bytes). The payload has a variable size ICMPv6 error messages which could partially contain the original ICMPv6 header [87, 89]. This is vulnerable to covert channel

```

08  ---  >>  Type ECHO 8
00  ---->>  CODE = 0
e8 eb ---->>  Checksum
0f 13  -----  >>  Identifier
00 01  ---  >>>  Sequence Number

```

Figure 2.11: The decoded ICMPv6 message format

Table 2.6: ICMPv6 Type network permitted messages [2]

Message Type	Value Sequence
Unallocated Error Messages	5-99, 102-126
Unallocated Information Messages	155-199, 202-254
Experimental Messages	100,101,200,201
Extension Type Numbers	127,255

use therefore the source and destination addresses of an ICMPv6 packet should be checked. Messages types 130, 131, 132, 143 for Multicast Listener Discovery must have link-local source addresses otherwise should be dropped. IANA has categorized ICMPv6 messages into two categories: Blocked and Permitted, see Table 2.6 for the permitted and Table 2.7 for the blocked messages [9].

Finally, Sohn *et al.* and Ulrich [12, 60] indicated that the unorganised content of the payload data in ICMPv6 may have different data type based on the messages types mentioned above. Additionally, the OS used also has its potential role. However, sometimes ICMPv6 packet transfers insignificant values or null values which can be an indication of existence of potential covert channels [37, 97].

Table 2.7: ICMPv6 Network dropped messages [2]

Message Type	Value Sequence
Destination Unreachable Messages	1
Packet Too Big Messages	2
Time Exceeded Messages	3
Parameter Problem Messages	4

2.7 The Need for New Detection Approach Against IPv6 Attacks

An intrusion detection system (IDS) monitors network traffic, looking for suspicious activity that could represent an attack or unauthorized access. They most commonly detect known threats based on defined rules or behavioural analysis through base-lining the network. A sophisticated attacker can bypass these techniques, so the need for more intelligent intrusion detection is increasing by the day. Researchers are attempting to apply machine learning techniques to this area [104].

Security management systems are developing continuously to protect networks and computers in all business sites physically and virtually. Current IDSs inspect all inbound and outbound network. However, one of the major challenges for any IDS or Intrusion Prevention System (IPS) is the detection of suspicious anomalies in fast network traffic. This is due to the variation of the pattern categories and because of the active threats such as Distributed Denial of Service (DDoS) attacks. One of the main objectives of a typical IDS or IPS system is to protect the three main essential elements of information security: Data Confidentiality, Data Integrity, and Data Availability [105].

Traditional applications can not fully protect networks and systems from increasingly sophisticated attacks like covert channels as well as DoS, specifically in IPv6 [106, 107]. Moreover, most of these traditional systems have been built based on such techniques suffering from high false positive and high false negative detection rates, in addition to the lack of continuously adapting to changing malicious behaviours in the past decade. Several Machine Learning (ML) techniques have been applied to the problem of intrusion detection in TCP/IPv4 in order to improve the accuracy and adaptability of detection. These techniques are often used to keep the attack knowledge bases up-to-date and comprehensive [107].

Wendzel and Zander *et al.* [74] investigated more than 109 covert channel techniques. They suggested the reduction of all techniques into 11 different patterns stating that 69.7% could be categorized into four different patterns due to the similarities of the attempted approaches. Wendzel *et al.* stated that using

traffic normalization against storage covert channel, as a technique to remove the ambiguities and breaking the policy in network traffic, could have side effects because these techniques are not effective any more [108]. Lewandowski *et al.* [17] used aggressive normalization to present a network-aware active warden against covert channels without creating an audit source of the attack instances.

The research community is aware about covert channel issue and considers it as a challenging task [109]. Due to the large amount of existing covert channel techniques, it is impossible to counter all covert channels in practise. Hence, in parallel to the amount of new developed techniques needed to tackle and counter such network security threats, implementing new approaches and techniques are necessary to mitigate some of these covert channels. This necessity is an aftermath of the incomplete implementation of IPv6.

2.8 State of the Art of Machine Learning Application in Intrusion Detection Systems

Machine Learning (ML), also known as Computational Learning Theory, is to understand the essence and principles of learning as a computational process that merges tools from computer science and statistics [110, 111]. It is a powerful tool used in search engines, medical diagnostics, face recognition, marketing, image recognition, traffic flow, and IP classification [112].

Researchers have been using two types of machine learning classifiers, single and hybrid, to promote anomaly intrusion systems [106, 113, 114, 115]. In this section, an overview about the state of the art of ML techniques implementation in IDS is highlighted.

(ML) algorithms optimize a performance criterion using sample data or past experience [111]. ML algorithms used in cryptography to develop algorithms for the eavesdropper [105], as its techniques can be used for boosting through creating a mechanism to extract an ultimate power out of a given algorithm. Recent studies argued that through significant work the impact of ML algorithms can have the ability to alter their input representation automatically using kernel functions, which are learned from data [21, 74, 105]. Therefore, a clear

understanding about the state of the art of ML techniques in Intrusion Detection Systems IDS is necessary.

In the early 1980's, James Anderson suggested the concept of intrusion detection in his seminal paper [116]. He presented a model to classify threats which could develop a security monitoring surveillance system. This model was based on anomaly detection in user behaviour. Later in 1986, Anderson presented several commercial IDS models based on statistics such as Markov chains, time-series, etc. [117]. Stanford Research Institute (SRI) added a few functions to the latest detection approach through monitoring user behaviour and detecting suspicious events [117]. Samha and Haystack [118] proposed another statistical anomaly-based IDS which targeted user and group based anomaly strategies. Forrest *et al.* [106] suggested an analogy between the immune system of a human and an intrusion detection system to analyse a program system called "sequences" in order to build a normal profile.

Debra Anderson *et al.* [119] and Cabrera *et al.* [120] suggested the same statistical methods for intrusion detection, both having used the same methods but deployed them in different way. Anderson *et al.*[119] suggested a comprehensive technique for intrusion detection system to perform real-time monitoring of any user on multiple on-line targeted computer. Meanwhile, the approach that Cabrera *et al.* suggested was a statistical traffic modelling to detect new attacks against computer networks.

Valdes *et al.* [121] suggested a developed version of an anomaly based IDS deploying Bayesian network to detect intrusion on traffic bursts. Kruegel *et al.* [117] suggested a novel multi-sensory fusion approached using Bayesian classifier in order to classify and suppress false alarm. Essentially, this classifier was result of an aggregated group of sensors to produce single alarm. Shyu *et la* [117] proposed an IDS anomaly based detection using principal components analysis (PCA), which was effective enough to reduce dimensionality of audited data.

Yeung *et al.* [122] implemented hidden Markov model in an anomaly based detection method to compute the possible likelihood of an observed sequence by using forward or backward algorithm to identify the anomalous.

Dickerson *et al.* [123] suggested a development of the Fuzzy Intrusion Recognition Engine (FIRE) using fuzzy logic in such a way that it generates the

fuzzy sets of data for each feature then use them to detect network attacks. Salih *et al.* [21] suggested a new hybrid approach using fuzzy logic and genetic algorithm to detect network storage covert channels in IPv6. This process analysed IPv6 and ICMPv6 header fields values and explains the viability of transferring strange instances which consequently indicate abnormal behaviours and possible covert channels.

Wenke and Xiang [124] proposed theoretical measures used in anomaly detection implementing examples of theoretic measures such as entropy, conditional entropy, information gain and information cost for anomaly detection. Ryan *et al.* [125] suggested a novel framework called Neural Network Intrusion Detector (NNID) using artificial neural network with supervised learning. The process was based on back propagation neural network; meanwhile, the supervised learning was trained in tasks to identify each instance. Most of anomaly detection methods with unsupervised features can create appropriate labels for all given instances automatically [79, 114, 121, 126, 127].

John and Langley [128] proposed a method to analyse the relationship between independent and dependent instances using conditional probability. They implemented Naïve Bayes classifier (NBC), based on a strong independence assumption with a quite simple structure [113]. Amor *et al.* [129] proved that Naïve Bayes classifiers offer more reasonable results, even with a simple structure. Strayer *et al.* [130] introduced a detection approach based on network behaviour and machine learning. They deployed in their framework several machine learning approaches: C4.5 decision tree, Naïve Bayes (NB) and Bayesian network classifier. These ML approaches were used in order to classify Internet Relay Chat (IRC) traffic flows as malicious or normal. The results were valued but the botnet still needed more development to include other types of attacks [131].

Experiments show that NB is very competent in classification tasks, but not so good in classifying User-to-Root (U2R) and Remote-to-Local (R2L) based attacks correctly. Apart from the network attacks against TCP/IP suite protocol, most studies have not covered covert channels attacks and security implications in IPv6 [21, 74, 113]. New attempts are needed to tackle and investigate covert channels using ML techniques in a integrated hybrid system in order to overcome most of

the limitations [21, 132].

2.9 Machine Learning Techniques

Machine Learning is used generally to find patterns in sample datasets through input of instances of individual sample of dataset with similar features. Hence, the output of ML will be the patterns and rules that ML learns from the sample dataset depending on the approach implemented. ML has a few types of techniques such as classification, clustering, association, numeric prediction [133]. However, it mainly handles regression and classification implementing multiple dependent and independent variables. In their statistical methods, ML uses various types of algorithms.

The most used ML techniques to classify intrusive and non-intrusive behaviours are shown in Figure 2.12. Machine learning is a particular branch of artificial intelligence that acquires knowledge from training data based on previously known facts. Machine learning mainly focuses on prediction and its techniques are classified into three broad categories [106, 134, 135]:

- **Supervised Learning:** It is known as classification. In supervised learning data, instances are labelled in the training phase which is needed to create testing data [134].
- **Unsupervised Learning:** This process deals with unlabelled instance of data. A prominent way for this learning technique is clustering. [134].
- **Reinforcement Learning:** In reinforcement learning process computer interacts with an environment to achieve a certain objective. A reinforcement approach can ask a user such as a domain expert to label an instance, which may be from a set of unlabelled instances [134].

2.9.1 Naïve Bayes Algorithm (NB)

NB was designed to be used for classification [129, 136]. It is a robust algorithm and deals with complex data, both symbolic and numeric. It is known to be

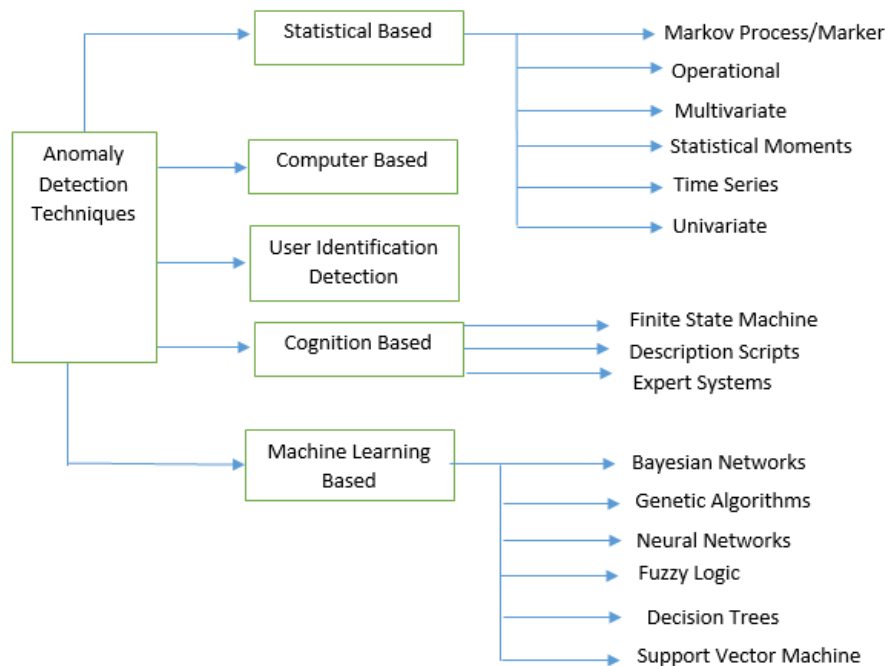


Figure 2.12: Machine Learning Techniques Classification [6]

reliable, fast, and easy to use, and interpret needs to be run only once. Naïve Bayes is specifically appropriate when dealing with the dimensionality of the high independent space (i.e., number of input variables). Naïve Bayes can perform more sophisticated classification along with other methods which models conditional distributions of the inputs however, other data mining techniques exist and perform differently to what NB does [137, 138].

NB classifiers consider any attribute of value given to a class as an independent value. This is called class-conditional independence which makes the computation very simple, elegant, and robustness [26, 55].

Naïve Bayes is a simple probabilistic classifier applying Bayes' theorem. This is accompanied with a level of powerful independence assumptions called conditional status of class independence, because it assumes any attribute's value of a class could be affected independently. NB is one of the fastest learning techniques and is able to check and examine almost all inputs during training [139, 140]. formula

2.1 explains the NB theorem, the standard Naïve Bayesian rule is:

$$P(c | x) = \frac{P(x | c) P(c)}{P(x)} \quad (2.1)$$

Where $P(c|x)$ = the posterior probability of class (target) given predictor (attribute), and $P(c)$ = is the prior probability of class. $P(x|c)$ = is the likelihood which is the probability of predictor given class and $P(x)$ = the prior probability of predictor.

The probability is calculated according to Bayes' rule as shown in formula 2.1. More explanations about data mining techniques and Naïve Theorem [133, 141] can be read.

2.9.2 Multinomial Naïve Bayes (MNB)

Multinomial Naïve Bayes implements the Naïve Bayes algorithm for multi-nomially distributed data. It is one of the two classic naive Bayes variants which can be used in text classification. Naïve Bayes classifier uses conditional independence of each of the features in the model, while Multinomial Naïve Bayes classifier is a specific instance of a Naïve Bayes classifier which uses a multinomial distribution for each of the features [112, 142].

Vectors can be used to set parameters of the distribution $\theta_y = (\theta_{y1}, \dots, \theta_{yn})$ for each class y , where n is the number of features and θ_{yi} is the conditional probability $P(x_i | y)$ of feature i appearing in a sample belonging to class y . The parameter $\hat{\theta}_y$ can be estimated by a smoothing version of maximum likelihood such as relative frequency counting:

$$\hat{\theta}_{yi} = \frac{N_{yi} + \alpha}{N_y + \alpha n} \quad (2.2)$$

where $N_{yi} = \sum_{x \in T} X_1$ is the times that the feature i appears in the class sample y in the training dataset T , and $N_y = \sum_{i=1}^{|T|} N_{yi}$ is the total count of all features for class y . Then, if the smoothing prior is used so $\alpha \geq 1$ will account the absent features in the learning instances and prevent zero probability in

further computations. If $\alpha = 1$ was set it will be called Laplace smoothing. If $\alpha < 1$ is used, it will be called Lidstone smoothing [135, 139]. Simply, in order to train Naïve Bayes for n-dimensional data with k classes, it needs to estimate

$$P(X_i|C_j)P(X_i|C_j) \text{ for each } 1 \leq i \leq n, 1 \leq i \leq n, 1 \leq j \leq k, 1 \leq j \leq k.$$

We can assume any probability distribution for any pair $(i, j)(i, j)$. It is better to assume variant attributes in the case $P(X_i|C_j1)P(X_i|C_j1)$ and continuous for $P(X_i|C_j2)P(X_i|C_j2)$. The multinomial distribution normally requires integer feature counts. Multinomial Naïve Bayes assumes multinomial distribution for all the pairs, which is a reasonable assumption in some cases [111, 138].

2.9.3 Decision Trees C4.5 and Feature Selection

Decision trees (DT) are among the well known machine learning techniques. C4.5 is a single classifier technique used to create a classifier for predicting the value of a target class for an unknown test instance, based on several already known instances. Through a sequence of decisions, a predicted instance is being classified by a decision tree [143]. A decision tree is a k-ary tree where each of the internal nodes specifies a test on some attributes from the input feature set used to represent the data. The basic algorithm for decision tree induction is the greedy algorithm which constructs decision tree in some sort of top-down recursive divide-and-conquer manner [136]. Decision tree is very popular as a single classifier because of its simplicity and easy implementation [134]. There are two types of decision trees: classification tree, with a range of symbolic class labels, and regression tree, with a range of numerically valued class labels [134, 143]. In decision trees C4.5, two essential phases should be ensured. First, based on a given training set, a decision tree is built which will consist of selecting the appropriate test attribute for each decision node and defining the class labelling for each leaf [134]. Second, in order to classify a new instance, the classification process starts from the root of the decision tree, then test an attribute specified by this node. The result of this test allows to move down the

tree branch relative to the attribute value of the given instance. This process will be repeated until a leaf is encountered. The instance will be classified in the same class as the one characterizing the reached leaf.

Quinlan has developed ID3 and C4.5 algorithms in order to ensure the construction of decision trees and their use for classification tasks [144]. Those are among the most popular ones. There are also some other types of decision trees [134, 143]. Feature selection is an attribute reduction process different from feature extraction, which ranks the existing attributes according to their predictive significance. [138, 145].

In network intrusion detection process feature extraction is processed by using advanced technique to ex-filtrate targeted features such as fields and their values, from the captured packets. Some applications of feature extraction are latent semantic analysis, data compression, data decomposition and projection, and pattern recognition [6]. Feature extraction can also be used to enhance the speed and effectiveness of supervised learning. Feature extraction can be used to extract the themes of a document collection, where documents are represented by a set of key words and their frequencies. Each theme (feature) is represented by a combination of keywords. The documents in the collection can then be expressed in terms of the discovered theme. [110, 146]. The accuracy of a classification model depends directly on the set of features provided in the training data.

There are other various ML techniques used in network anomaly detection such as Genetic algorithm [21, 126, 134] and Support Vector Machine (SVM) [6, 112, 147] that have optimal results in some cases. However, Multinomial Naïve Bayes is more suitable for text based classification.

2.10 Summary

In this chapter, a brief comparison between IPv4 and IPv6 has been presented. The security issues of the IPv6 also have been explained. The occurrences of various security attacks exploiting the design vulnerabilities have also been discussed. Most of the IPv6 security issues have been addressed. The security status of IPv6 protocol is quite unstable and exploitable according to the previous facts analysis and technical investigation. IPv6 has not solved all IPv4

inherited security vulnerabilities. There are still many ongoing processes to mitigate these security vulnerabilities, running focusing on the recent threat incidents against legitimate targets in the business and official networks.

Covert channel is one of the embedded or encrypted techniques used in data hiding to perform attacks or in/exfiltrate classified information from victims' vulnerable systems. Exploiting header fields and payloads of IPv6 is not obvious due to the complexity of the protocol design and its incomplete implementation on current operating systems. Few researchers have successfully analysed and investigated the behavioural storage covert channel in terms of security vulnerabilities to eliminate the violations of privacy and network security. The state of the art in covert channel detection frameworks has been discussed. Different ML techniques have been highlighted in terms of their implementations in IDS and network security systems. Furthermore, an overview about the state of the art in ML application in IDS has been discussed as well. The unavailability of similar approach to what is suggested in this thesis has been elaborated and other different approaches in the domain problem have been discussed. ML has been used to tackle some issues of network security, whereas, hybrid approaches to tackling and eliminating hidden communication attacks have not been recorded yet in IPv6.

Chapter 3

Proposed Detection Framework

3.1 Introduction

One of the most significant limitations in IPv4 is the address depletion due to the massive growth of Internet users in the past decade [2, 3]. In addition to this fact, there are other security vulnerabilities as mentioned in Chapter 2. Basically, the security threats and vulnerabilities in IPv4 and IPv6 are the aftermath of restrictions parameters limitations of its design infrastructure; in other words, the inherited weaknesses in TCP/IPv4 protocol design led to less development and limited modifications in IPv6 [17, 26]. Covert channel phenomena are caused by two categories: design oversights and inherent weaknesses from the previous system, as discussed in Sections 1.1 and 2.4. Most researchers [11, 26, 74] agreed and explained that covert channels should be identified and confirmed prior to initiate any detection or prevention process. When a covert channel has been identified, the basic countermeasures will be taken into account. Despite the fact that previous researchers have identified them in their approaches, the continuous changing of the protocol fields in the Request for Comments (RFCs) implementation has produced nearly more than 35 RFCs and standards until preparing this thesis (for a list of investigated RFCs see Appendix A.2).

In order to obtain robust and constant results from the research domain and the suggested hypothesis, a controlled network environment with simulation network design needs to be created. This will be explained in the experiment

part in Section 4.2. Using covert communication is considered a sensitive subject in security threats against IPv4 and IPv6. The existing frameworks [20, 30, 148, 149] have categorized abnormal and unknown attack detection methods into three main categories:

- Signature based detection: is done through building and updating a signature database, informing the network administrator about any recognized signatures.
- Protocol based detection: focuses on protocol anomalies and violations of privacy and security policies; the operator will be notified.
- Behavioural based detection: is done through creating a behaviour user profile and using statistical method to detect if the data stream has a suspicious status.

In this thesis, a suggested security system which handles large data flow volume will be presented and discussed. Detecting suspicious and hidden data carrier using the new advanced technology against legitimate targets is a vital demand. However, the current security system demands throughout building the conceptual framework have been considered. New approaches are needed to tackle and eliminate such advanced security threats against IPv6. A novel Network Intelligent Heuristic Algorithm (NIHA) is suggested into a security system to detect and classify covert channels. This approach has been hybridized with an enhanced Multinomial Naïve Bayes classifier in order to improve detection performance, giving a higher accuracy positive rate and a low false negative rate.

The proposed framework consists of five modules as follows:

1. Capturing raw data module.
2. Packet filtering module.
3. Data pre-processing module.
4. Detection and classification module (Covert channel analyser).

5. Issue verdict module.

Hierarchically, full details of all modules can be organized and discussed in three integrated phases:

- **Phase One:** provides an overview of each module and its functionality starting from Section 3.5. This part gives the basic details of the approach through discussing details about each stage accordingly with some examples.
- **Phase Two:** provides an explanation of building the classifier module and its algorithms, starting from Section 3.6.
- **Phase Three:** provides in-depth details about the implementation of all modules through creation of the overall algorithms and its engines for each stage, starting from Section 3.7. This part provides detailed algorithmic expression of each stage in an interrelated order with explicit examples for each attribute.

This chapter will be organized as follows: Section 3.2 explains the main scenario; Section 3.3 discusses the probabilistic model; Section 3.4 discusses the main issues in data aggregation process and data collection methods; Section 3.5 explains the stages of the new suggested model; Section 3.6 discusses building the classifier; Section 3.7 presents the implementation of the new algorithm "HeuBNet6", its modes and detection engines. Finally, Section 3.8 summarizes the chapter.

3.2 The Main Scenario

In consideration of the different attacks and vulnerabilities mentioned in the previous chapter, a further step was taken to identify main scenario used for the overall design which is the de-facto standard model for covert channel communication. It is the classical prisoners' problem [26, 150]. Two people have been thrown into prison and intend to escape. In order to let this happen, they should agree to a plan and need to communicate. However, all their communication messages are monitored by the warden. To extend this scenario

3. Proposed Detection Framework

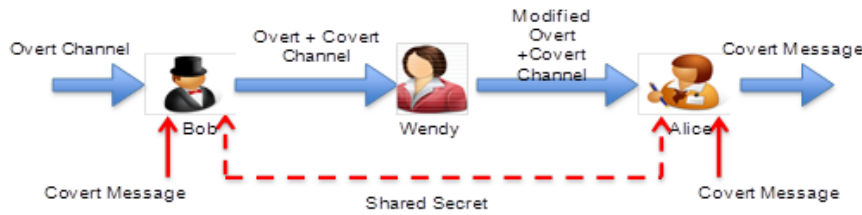


Figure 3.1: The prisoners' problem with communication monitored by warden [7]

as shown in Figure 3.1, the two prisoners called Bob and Alice are two agents who wish to communicate covertly. Exploiting the existing communication path corresponds to two random communicating processes: the sender and the receiver. Wendy is a warden located somewhere on the communication path hence monitoring all possible messages exchanged by Alice and Bob.

The dotted double arrows indicate that Alice and Bob could either act as sender and receiver, or could modify messages in transit. Wendy thinks that these situations are indistinguishable. Meanwhile, she must approve all communication messages between both. In order that plan to succeed they need to embed some details in the protocol header and exploit the vulnerable design format such as Traffic class, flow label, and the unused spaces along with the reserved fields and pass it to each other.

Wendy has few options. As an active warden, she can modify the network packets between them which is not our case here. As a passive warden, she should monitor the communication messages, analyse the attacks in the traffic, detect the anomaly behaviour behind the header fields of the protocol, create instances of each attack and log them in order to detect future unknown similar attacks. This is to mitigate the security vulnerabilities.

In the suggested model, it is assumed that the prisoners are communicating together in an unreliable environment with a passive warden. Some packets will be lost or duplicated. Hence, in our hypothesis TCP protocol will be partially included due to the minimization of the domain problem and research question criteria. The covert channels need to be:

- Indistinguishable: Wendy (a passive warden) will not be able to detect the

existence of the data hidden in the packets sent by Bob from Alice.

- Reliable: Alice needs some indications or signs to state that Bob's packets have arrived.

3.3 Probabilistic Approach

Deep packet inspection through layers 2 to 7 in Open System Interconnection (OSI) reference model offered a promising level for new methods to classify network traffic. The approach depends on signature based comparison of packet flows, heuristic, statistical, or anomaly-based techniques. The motivation was to develop packet data inspection in the same application layer that performs the filtering instead of using different systems to achieve a faster detection process. Apparently, this development has solved some issues and left other new unsolved issues behind, because it considered many complex tasks in order to be effectively successful.

Lewandowski [11] implemented aggressive normalization in his research, where audit data and the original attack instances were normalized in such a way when attack instances were detected, their features and attributes were normalized to the idle value (i.e. from 0 – *FFFFF*). Active warden approach has a few limitations which will be discussed in Section 5.6. Conversely, in this thesis, we suggest a hybrid active warden presented as a multi-threaded process for the classification model using NIHA. Furthermore, enhanced Multinomial Naïve Bayes Algorithm application is suggested. This is a novel approach as previously no attempt has been referenced to storage covert channel detection in IPv6 [46, 132]. Potentially, new approaches are needed to detect storage covert channels in IPv6, particularly using a machine learning technique such as MNB in response to the novel vulnerabilities arising every day. Using supervised machine learning techniques to tackle such anomaly in IPv6 will add a new route of cutting edge solutions for security systems. Most of the existing methods [5, 17, 20, 74] have the following drawbacks:

- Using complicated algorithms to detect encrypted covert channels.
- Creating traffic congestion while processing.

- Time consuming in on-line detection.
- Few parameters are considered in dealing with covert channels.

According to previous investigations in Sections 2.5 and 2.6 the following fields' values have the highest indication tuples to identify IPv6 and ICMPv6 covert channel attacks: Traffic Class, Flow Label, Hop Limit, Payload Length, Next Header, Source Address, ICMPv6 Type and Code.

These types of covert channels could be discovered using probabilistic approach for the occurrence of covert channels in the ip tuples. A probabilistic model was used to distinguish between scanning sources and normal users [33]. When it is operated, it gives a particular class membership. The probabilistic approach in modelling network security systems is effective and flexible to capture relevant aspects of the problem domain, and flexibility can be achieved by allowing models to have sufficient parameters [151]. The fundamental idea in Bayesian modelling is to use the mathematics of probability theory to represent and manipulate all forms of uncertainty in the model. This is a surprisingly simple yet powerful idea. The feature that distinguishes NBA from other algorithms in classification is that there are only two rules of probability theory needed to be remembered: the sum rule and the product rule. In this case, MNBC will predict the covert channel class type (Yes/No) as a given class for calculating the posterior probability together with the attributes existing in the received packet. However, the MNBC expects numeric values for the attributes in order to calculate the probabilities. Therefore, a lookup table depending on every possible set up value in the targeted field according to RFC 2460 will be created to map the possible states of the attributes (see Table 3.1). This look up table will start with Zero if the attribute is not present in the captured packet, to N, the total number of values or categorisations the attribute can hold. Furthermore, the continuous development with the RFCs versions did create some technical problems during model implementation.

3.4 Data Aggregation Process

Researchers [5, 17, 145, 147, 152, 153] have suggested various approaches and frameworks to detect different attacks against IPv4 and IPv6. Some of them have

Table 3.1: ARFF Header format for Validation Process

@Relation	NIHATest	
@Attribute	Class	{1, 2}
@Data	Packet	String

dealt with covert channel problems as discussed in Section 2.3 and 2.8. This thesis suggests a different approach to what Lucena *et al* [10] and Lewandowski [11] *et al* have suggested to eliminate and mitigate covert channels in IPv6. A discussion about the differences between similar approaches [5, 17, 153] to the suggested method in this thesis will be presented in Section 5.6.

To capture streaming data in an active operation mode containing unknown attacks or covert channels is a challenging task. This is because of the ever-increasing traffic throughput of the heterogeneous and non-stationary data and the mass communications in the network interconnections paradigm. Most of the IDS mechanisms depend on off-line captured data and the evaluation part stimulated from certain amount of data taken out from the training dataset.

The accuracy of such detection methods is quite high [26] depending on the quality of off-line stored data. Meanwhile, the on-line accuracy has been improved relatively to limited targeted attacks in some of the application layer protocols. There are some "micro" attacks using the protocol's header fields to perform sensitive and complicated attacks. These incidents can not be detected unless a combination of techniques is deployed and effectively used against such attacks [17]. On the other hand, the massive data also contributes to the complicated real-time detection process. Furthermore, the apparent reason of this drawback is that the IDSs focus only on the network packets rather than analysing host events. This vital issue has caused low performance and limited accessibility into the protected system domain [26].

IPv6 firewalls and IDS have not been fully implemented on the current operating systems yet. Most of the current web servers use dual stack mechanism, tunnelling IPv6 through IPv4 and other techniques [103]. Hence, the latest statistical IPv6 usage has barely reached 16.04%. These issues have been discussed in Sections 2.5 and 2.6, in addition to the details of characteristics of

such attacks in Sections 3.1 and 3.2. The distinctive issues of IPv6 are the incomplete process and the inherited security vulnerabilities in its design. This current status of the protocol potentially needs new approaches to protect the network from harmful attacks. The encapsulation protocols and the dual stack are suffering from such security issues as well [5].

Not all IDS models are trained to be deployed over computer networks deceiving high performance in off-line mode. In on-line mode operation, latest IDS models function perfectly to classify network traffic. However, this was not ideal for limited targets in application layer within TCP/IP version 4. This issue persists for IPv6 with the current relevant anomaly detection and anti spy tools. Substantially, most of the IDS models face difficulties with massive data flow which leads to sophisticated and non-stationary network security problems [103, 153, 154].

Massive data stream is the major inevitable issue against professional performance for IDS and similar security system such as covert channel detection. In order to overcome these unsolved problems in IDSs, new techniques and approaches should be used to convert huge data flow to connection attributes. In real-time analysis, all attack instances and data types should be dealt with to create testing and training data.

3.4.1 Data Collection Method

Numerous network security tools have been used to study IPv4 covert channel instances. However, not so many tools exist to analyse and study covert channels in IPv6 [17, 74]. Most of them depend on generated benchmark data such as Network Simulation Language Knowledge Discovery Dataset (NSL-KDD). Due to important ethical regulations, controls and acts compliance with Data Protection Act 1998, it is illegal to perform live attack simulation on our university network systems. Instead, a controlled computer network will be created on Oracle virtual application GNS3.

Dealing with massive data is vital for this approach to tackle covert channel. For this necessity, Netfilter was used for real-time data capture. Netfilter/IpTables is the kernel extension of Linux which hooks the Ip6tables firewall functionalities. The method to capture data from the suggested network design depends on the

specified mode of the testing plan. Linux based firewall Netfilter is used to forward the traffic prior to the routing state as shown in Figures 3.2 and 3.3.

The data was collected using the main attack simulation method for different attacks with the help of the following tools:

- The Hacker Choice (THC): An IPv6 vulnerability security testing tool [50].
- Scapy client/server: An attack simulation tool created to perform some types of network attacks on a suggested testing environment [46].
- Modification of Buchanan's [27] tool which is written in C Language.
- Modified Socket Programming Tools [46].
- IPv6ToolKit [50].

3.5 Suggested Model Stages

This section discusses the suggested framework modules: raw data capturing module, filtering and data analysis module, data pre-processing, detection and classification module, and decision module. Each module is labelled in a sequence order. The modules are divided into two main activity spaces as shown in th Figure 3.2. The raw data capturing module is in the kernel space and the rest modules are in the user space.

3.5.1 Raw Data Stage Capturing

This is the first stage to monitor and capture raw data from the network traffic simulated attacks. There are different techniques to capture traffics, either by attack simulation which creates raw packets sent over the wired or wireless network, or capturing live data using sniffer tools. Most of them depend on generated benchmark data such as Network Simulation Language Knowledge Discovery Dataset (NSL-KDD) which is a newer version of KDD'99. This version of NSL-KDD data is modified and altered after certain operations as discussed below [153], therefore, this benchmark data will not be used in any sort of

3. Proposed Detection Framework

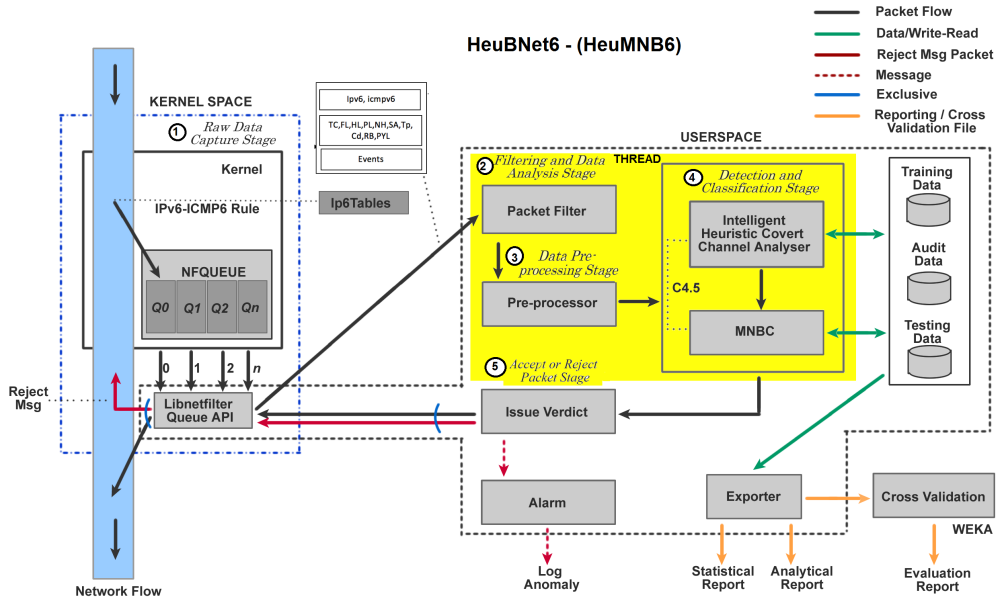


Figure 3.2: Proposed Covert Channel Behavioural Detection Model

evaluation of validation process. Moreover, it is not compatible with the suggested model.

With regards to NSL-KDD benchmark data, the following elements have been considered during the initial steps in raw data capturing phase:

1. There will be no secondary synthetic data to evaluate or verify this approach; instead, the primary data will be created during the development process. This data is essential and vital to be validated on other applications and validation techniques later.
2. In order to verify the preliminary results data types in this project using NSL-KDD, previously this benchmark data was tested and evaluated with similar format in the primary stages of the project [46, 132].
3. The benchmark data NSL-KDD is a result of testing on IPv4 and not compatible with IPv6.

Furthermore, the data format presented in this thesis is different. This is due to the fact that researchers at the University of New Brunswick with Canadian

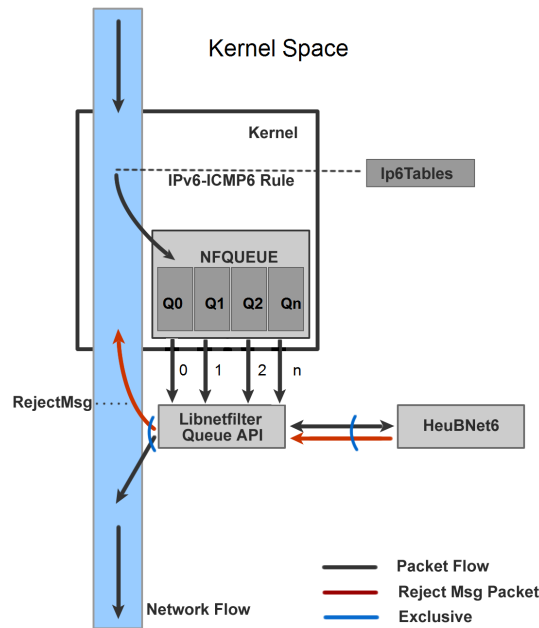


Figure 3.3: Linux Netfilter traffic netflow

Institute for Information Technology presented two main issues in KDD'99. These problems affected the evaluated systems' performance in addition to the very poor evaluation of anomaly detection approaches [145, 153].

Netfilter¹ as shown in Figure 3.3 was chosen after thorough investigation on similar network capturing tools. With respect to capturing tools, there exist a number of tools such as Hping3, Netcat, and WinPcap. However, due to some performance considerations and inherent problems, the idea to select these tools as capturing tools in the proposed system was weak due to some serious issues: the application layer sniffing tools are used for analysis purposes only and they possess high volume processing problems etc., according to the suggested network topology.

Initially, in the Kernel Space, the blue long box represents the network stream and the black arrow presents data flow which contains the simulated attack raw

¹Netfilter is a packet filtering software framework for Linux 2.4.x and later kernel series. The Iptables firewall is used together with this tool. Iptables enables packet filtering, network address [and port] translation (NA[P]T) and mangling of packets. It is the re-designed and heavily improved successor of the previous Linux 2.2.x ipchains and Linux 2.0.x ipfwadm systems [155].

packets. Raw data was captured out of attack simulation. This is done on Linux OS based Netfilter and Libnetfilter API. Multiple arrows are used to represent the packets' queues. This process is vital due to the traffic speed holding large size throughput buffer reaching between 10 Gbits to 40 Gbits. NFQUEUE is used in IP6table rule chain to queue (e.g. q0, q1, q2,..qn) packets in the kernel. Additionally, it deals with such speed by the suggested multi-threaded capability which is represented in multiple black arrows in sequence starting from 0, 1, 2,...,n. Each arrow as shown in Figure 3.3 represents a thread which uses a core of the CPU and 1GB RAM in the computer. The arrows are directing towards the Libnetfilter API to process the data flow in order to deal with voluminous incoming packets simultaneously. The details are implemented in Algorithm 6.

3.5.2 Filtering and Data Analysis Stage

In the second stage, the input data is filtered by performing field selection to choose the targeted fields and their values. The data flow is represented in a long black arrow headed towards the packet filter process as shown in Figure 3.2. Packet data from the IPv6, ICMPv6 headers and the TCP upper layer protocol is shown in the box between the user space and the kernel space by a dotted line holding to the header fields that will be filtered. TCP was partially examined for analysing covert channel fragmented packets in the extension header whereas UDP was not considered within the scope of this thesis due to the unreliability features in the protocol [1]. In this stage, network packet data (in bytes or Hexadecimal format) is transformed into human-readable format such as ASCII using Network-to-Host translation (NTOHL) script for field values and INET_NTOP script for IPv6 address fields. An example is given below and the details of technical and implementation process are explained in Algorithm 3.

Figure 3.4 is an initial example which explains converting between binary to ASCII format using 7 bits code with either odd or even parity bit such as $X_6X_5X_4X_3X_2X_1X_0$, where X_i is either 0 or 1, $i = 0, 1, \dots, 6$.

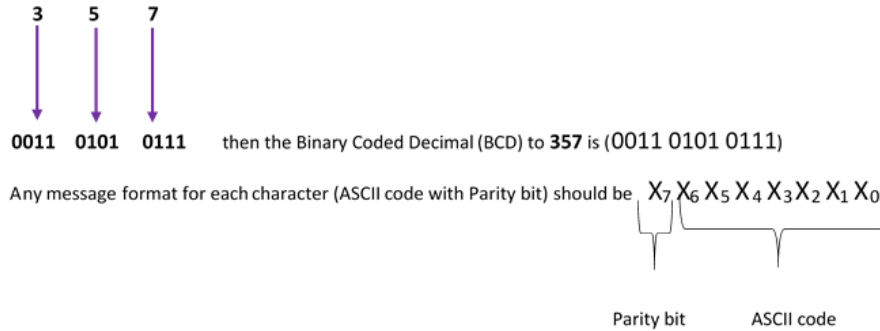


Figure 3.4: Filtering Data Type from Captured Packets

3.5.3 Data Pre-processing Stage

In the third stage, the input raw network data went through field selection, dissected and normalized to remove data complexity. This stage is vital due to the sensitivity of the feature selection process prior to detection and classification process since data redundancies, unwanted protocols and header fields should be removed from each captured and filtered packet. Practically, only the relevant and selected data needs to be handled. The minimised captured data should flow into the covert channel analyser, which is called Network Intelligent Heuristic Algorithm (NIHA). An example is shown in Figure 3.5 and it will be explained in Section 3.7. All fields in IPv6 packets must be checked for covert channels during the process of converting nominal values to Naïve Bayes frequency table for HeuBNet6 in live capturing mode. NIHA transforms the nominal values to pairs of numeric-ID representation and frequency of occurrence such as $253:3$ or $\{620:3 \ 34:1 \ 16435:2 \ 67343:1\}$ as expected by the HeuBNet6 MNB Engine. All known features' nominal values have their NB feature IDs recorded in a class map with the following format:

1. Nominal ID: is a nominal value representing a known feature (IPv6 Packet field value).
2. NB ID: is a numeric Naïve Bayes ID of the feature.
3. Class ID: is a classification ID of the feature's value (1 means anomaly and

2 means normal).

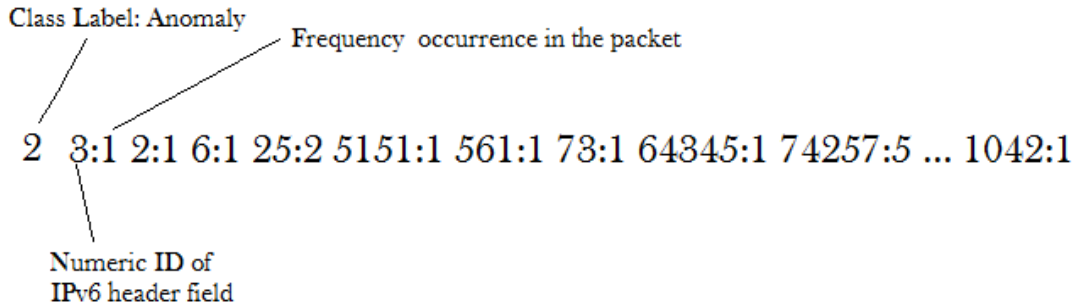


Figure 3.5: Example of Header Field Conversion to NB-Class Instances

Table 3.2: ARFF Header format for Validation Process

@Relation	NIHATest	
@Attribute	Class	{1, 2}
@Data	Packet	String

3.5.4 Data Training Stage

The captured input packets are filtered, transformed and discretised then streamed into the pre-processing module NIHA in order to create the datasets. NIHA then saves the datasets (training, audit and testing datasets) into an SQL database as shown by two double headed green arrows from and to the detection and classification module in Figure 3.3. The output header sample data as shown in Table 3.2 is taken from the database will form several Attribute-Relation File Format (ARFF) files which will be used in WEKA 3.7. Packet data as shown in Table 3.4 will be processed through Discretization and Transformation processes to form a dual type of *var_char* format as shown in Table 3.3. This is by using actual-value to nominal-value conversion during a feature's covert channel detection process. The nominal value of the resultant string will be converted to

NB format afterwards as shown in Figure 3.5 as an input into the suggested enhanced Multinomial Naïve Bayes classifier. A further advantage gained in dealing with massive data size issues in the use of Naïve Bayes classifier is that it deals with symbolic and numeric data and considers these attributes as independent values in classifying process. The expected effect will be the reduction of datasets size which will be processed, as these datasets are represented in a more convenient and economical format. Details of how captured massive data is dealt with are presented in Section 4.3.

Table 3.3: Feature Nominal-NB-Class Map

Attributes	Occurrence	Class
HopL_INC	1	1
HopL_DEC	2	1
HopL_UNC	3	2
PLen_INC	4	1
PLen_DEC	5	1
PLen_UNC	6	2
Extra_CovHdr	7	1
No_Extra_CovHdr,	8	2
Type1_Code0	9	2
Type1_Code1	10	1
....		
Word_10306	75605	1

3.5.5 Detection and Classification Stage

In the fourth stage, after the conversion and transformation process of the data, this module receives a set of data in order to label and process them into covert channel's analyser. Detection methods suffer from incapability to depict unknown (new) attacks carrying new updated signatures due to the offline focused training dataset. Furthermore, one of the vital elements causing an obvious degradation to most IDSs is that these IDSs are not targeting the network packets headers values rather than focusing on the payloads or the packets contents transferred from one node to another. Eventually, an issue of incompatibility in most IDSs for the IPv6 protocol raises. Moreover, IDSs suffer

Table 3.4: Nominal values to Naïve Bayes Format

Header	Selected Field	Input Value	Nominal Values	NB Format
Pseudo	Traffic Class	89 90	T_TC, F_TC	1233:1 1234:1
Pseudo	Flow Label	<i>src,port,dst,port,flow label</i> 2000::3 2001::19	T_FL F_FL	65:1 66:1
Pseudo	Hop Limit	456 457 458	HopL_INC, HopL_DEC, HopL_UNC	345:1 346:1 347:1
Pseudo	Payload Length	433 434 435	PLen_INC, PLen_DEC, PLen_UNC	100:1 101:1 102:1
Pseudo	Next Header	6 (frag) 22(telnet)	Extra_CovHdr, No_Extra_CovHdr	85:1 86:1
Pseudo	Source Address	2001::7 , 2001::13, 2001::15	Node_1, Node_2, Node_3	233:1 234:1 235:1
ICMPv6	Type & Code	1,0 255,255	Type 1_Code0 Type 255_Code255	45.1 65.378:1

from struggling to adapt and cope with heterogeneous networks [153]. The implementation process and examples are given in Algorithm 3. This fourth module can be divided into two sub-processes.

3.5.5.1 Covert Channel Analyser

This module is the first part of the fourth stage. It receives the packet data from the previous process in string format *@Data* field as shown in Table 3.2. The module is the main part of the multi-security detection system Network Intelligent Heuristic Algorithm (NIHA, the covert channel analyser). This algorithm will process to:

1. Analyse, detect and classify covert channels in header features.
2. Transform the values of each attributes and its subset values to nominal value after detecting the existence of a covert channel in the attribute.
3. Create novel primary training dataset for IPv6 (attack instances) and label them.

4. Store new attack types into the database in order to be used for live classification by the MNBC and validation stage later.
5. Make decisions upon the current and previous instances to detect new (unknown) attacks in the future.

3.5.5.2 Multinomial Naïve Bayes Classifier (MNBC)

This module is the second part of the fourth stage of the approach. This part operates as an extra security process along with NIHA. It serves to provide a multi-layer security system which offers high detection rates by using a proven probabilistic approach classifier. The training dataset for this stage consists of decoded packets transformed into ARFF format. The flow is processed by a feature selection technique using C4.5 decision trees and information gain. This is shown in black dotted line between covert channel analyser and the MNBC process in Figure 3.2. This is to minimize the redundancies and the least wanted features are taken off. In this stage, the most informative attributes in the projected data will be introduced into Naïve classifier to create the data model of covert channels and allowing this model to predict future anomalies in IPv6 network traffic.

3.5.6 Issue Verdict Stage

In the fifth stage of the framework, the decision process is activated as far as the labelling packets have been dealt with by the covert channels analyser. The data flow is represented in a black arrow pointed to the issue verdict process as well as the decision process in the system as shown in Figure 3.2. The following actions will be taken:

1. Rising a message of an anomaly detection by the classifiers to decide whether to allow the packet through or to drop it.
2. Generating alarm log entries and statistical report out of received data, represented in a dotted red arrow pointing downward to the alarm entity as shown in Figure 3.2. This message includes:
 - Predicting attack behaviours.

- Recording to a log of the anomalies.
- Offering auditing and event documentation procedures.
- Alerting the system administrator by creating an alarm, offering a full set of rules to deal with any unknown attacks which will be identified in the future.
- Accepting or rejecting packets through setting a flag in the message passed to the Libnetfilter API.

As shown in Figure 3.2, the long black arrow heading back to Libnetfilter API indicates the acceptance of the packet, and the long red arrow next to the black arrow indicates the rejected packet sent back to the flow. The blue line across both arrows indicates that only one of the processes will occur at a time. Thus, a packet can not be accepted and rejected simultaneously.

3.5.7 Exporter and Validation Stage

This is the last step of the proposed model. In this step, verifying the accuracy of the classifiers is performed by using a new format of training and testing data ARFF files for cross validation against WEKA (the industrial standard tool for packet classification and data mining). This process is represented in an orange arrow pointed to the Cross Validation process as shown in Figure 3.2.

The ARFF format file consists of three headers: relation, attribute and data. Each word has the prefix of @ as shown in Table 3.5. The header section of the ARFF file contains a list of the attributes (data in columns), and their types. Thus, Table 3.5 has the @Relation as **NIHATest**, the @attribute **Class** represents the detection classes (1 for Normal and 2 for Anomaly), and @attribute **Packet** represents the nominal representation of values in the IPv6 header fields, ICMPv6 header and extension headers. The @data section in Table 3.6 is the normal transformed data from the network simulation language.

The Exporter produces two types of reports: statistical and analytical reports. The reports flow is represented in orange arrows as outcome from the system as shown in Figure 3.2. Details about these two reports' contents are explained in Section 5.6.

Table 3.5: ARFF Header format for Validation process

@Relation	NIHATest	
@Attribute	Class	{1, 2}
@Data	Packet	String

Table 3.6: Data format in ARFF output file prior to detection process

@Data						
2	F_TC	F_FC	HopL_INC	Plen_INC	No_Extra_CovHdr	...
2	F_TC	F_FC	HopL_UNC	Plen_DEC	Extra_CovHdr	...
1	F_TC	F_FC	HopL_UNC	Plen_UNC	No_Extra_CovHdr	...

3.6 Naïve Bayes Classifier

NBC is intended to improve the performance of the classification process. The Posterior Probability equation 2.1 and the conditional independence assumption of Naïve Bayes classification is used to find the class of each variable. This process uses the set of IPv6 fields in Table 3.7 as inputs to the classifier using information gain to measure the quality of the classified data.

3.6.1 Information Gain and Gain Ratio

Information gain (IG) [156] measures the amount of information in bits about the class prediction, when the existed attribute is the only information available

Table 3.7: Identified Covert Channels in IPV6 Header Fields [5]

ID	Field	Covert Channel	Bandwidth
1	Traffic Class	Set a false Traffic Class	8 bits/packet
2	Flow Label	Set a false Flow Label	20 bits/packet
3	Paylaod Length	Increase value to insert extra data	Various
4	Next Header	Set a valid value to add an extra extension header	Various
5	Hop limit	Increase/decrease value	~1 bit/packet
6	Source Address	Set a false address	16 bits/packet
7	ICMPV6_Type	Set false Type value	8 bits/packet
8	UCMPv6_Code	Set false Code value	8 bits/packet

about a feature which corresponds to class distribution. Concretely, it measures the expected reduction in entropy (uncertainty associated with a random feature). In order to select the best test attributes, the entropy measurement needs to be worked out to calculate the purity in an arbitrary collection of examples. Let S be a set of training set samples with their corresponding labels. Suppose they are m classes and the training set contains s_i of class I and s is the total number of the samples in the training set. The expected information needed in order to classify a given attribute is calculated by:

$$I(S_1, S_2, \dots, S_m) = - \sum_{k=1}^m \frac{S_i}{S} \log_2 \frac{S_i}{S} \quad (3.1)$$

A feature A with values (a_1, a_2, \dots, a_v) can divide the training set into v subsets containing the samples S into (S_1, S_2, \dots, S_v) subsets where S_j is the subset which has the value a_j for feature A . S_j contain s_{ij} samples of class i . The entropy of the feature A is

$$E(A) = \sum_{j=1}^v \frac{(S_{1j} + \dots + S_{mj})}{S} \times I(S_{1j}, \dots, S_{mj}) \quad (3.2)$$

Then the information gain for A would be calculated by:

$$InformationGain(A) = I(S_1, S_2, \dots, S_m) - E(A) \quad (3.3)$$

In the given assumptions, information gain is calculated for class labels by employing a numerical discrimination for each class [157]. If the class has the same label in a specified dataset instance, it is considered as in-class instance and if the class has a different label, it will be considered as an out-class instance. Eventually, information gain for each class should be calculated; therefore, this will denote how well the feature can discriminate the given class such as normal or anomaly from other classes [136].

To work out the information ratio, the information held by any attribute needs to be split into its supposed many values. The attribute with the highest information gain is chosen to test the current node. However, information gain approach has some problems: it is biased towards tests with many outcomes

specifically when an attribute has a large number of values; it produces a large number of partitions (1 tuple per partition) with each resulting partition S having $\text{Info}(S) = 0$, and lastly, the information gain is maximized. C4.5 techniques need to be maximized with another alternative measurement called Information Gain Ratio (IGR), which maximizes the probabilities of considering each value of any attribute no matter how many values there are. Gain ratio applies a kind of normalization to information gain using a split information value. The split of the information takes into account any attribute having many values as shown below:

$$\text{SplitInformation}(A) = - \sum_{j=1}^v \frac{S_j}{S} \log_2 \frac{S_j}{S} \quad (3.4)$$

As explained in 3.1, the gain ratio can be worked out as below:

$$\text{GainRatio}(A) = \frac{\text{InformationGain}(A)}{\text{SplitInformation}(A)} \quad (3.5)$$

A ranking table is created for the gain ratios calculated for each of the attributes in the packet which determines which attributes are more prone to covert attacks. The attributes with the least gain ratio could be disregarded to obtain a smaller set of attributes to perform the packets classification.

Traffic Class is one of the critical fields in IPv6 due to holding false and true values which indicates either an anomaly packet or a normal packet. In the following example, this header field's attribute will be computed applying the techniques above to find the gain ratio and the class type. For Traffic Class category "True Class" distribution there are four Yes and three No. For "False Class" category there are three Yes and zero No, using equation 3.4.

$$\begin{aligned} \text{SplitInfoTrafficClass}(S) &= \frac{3}{10} * \left(- \frac{3}{3} \log_2 \left(\frac{3}{3} \right) \right) \\ &+ \frac{7}{10} * \left(- \frac{4}{7} \log_2 \left(\frac{4}{7} \right) - \frac{3}{7} \log_2 \left(\frac{3}{7} \right) \right) = \mathbf{0.846} \end{aligned} \quad (3.6)$$

The Gain ratio of **Traffic Class** is **0.846**.

3.7 The Novel HeuBNet6 System Algorithms

This is the third phase to implement the suggested approach. The HeuMNB6-Net has been shown in Figure 3.2. In order to have a unique name introducing a novel technique to IPv6 covert channel detection mechanisms it will be called HeuBNet6. HeuBNet6 features two integrated active classifiers for high detection accuracy. The first is called IHA or adding a prefix "Network" to IHA making it NIHA as a novel extensible mechanism for IPv6-RFC based benchmark dataset creator doubling as a covert channel detector. The second one is called Multinomial Naïve Bayes Classifier (MNBC) for near-real time probabilistic covert channel classification.

The HeuBNet6 is modelled on the IPv6 RFC's 2460 to 6294 [36] (a list of all investigated RFCs is presented in Appendix A.2). HeuBNet6 is designed to achieve the highest detection rate for covert channels in network packets. A semi-high level algorithm for the HeuBNet6 system is described below. This algorithm discusses the artefacts of the system starting from data creation stage through to the cross validation stage of results. For cross validation stage, hold out method is prescribed using WEKA [158] in Section 5.1.

The algorithms focus on presenting a novel *modus apparandi* for creating new datasets for IPv6 storage covert channel detection. When writing this thesis, there does not exist a benchmark dataset for IPv6 similar to NSL-KDD Datasets which is merely for IPv4. HeuBNet6 is designed to incorporate data acquisition for new training datasets as described in Section 3.5 and to offer a high rate of detection accuracy through an enhanced version of Naïve Bayes Classifier using ML techniques such as C4.5 and Information Gain techniques. In this approach, primary data is created through simulation of different covert channel attacks on the suggested IPv6 LAN design. This is performed through using vulnerability security tools written in C, Python and C++ programming languages to simulate these attacks. Different attacks are simulated using different covert data in the IPv6 headers. Table 3.4 shows the pre-processed output data format used for NB

classification and Table 3.8 shows the classification classes format.

The new approach presented through this research avoids complex and time consuming methods such as Support Vector Machines (SVM) [12] or other algorithms. The development occurs through utilising Multinomial Naïve Bayes coupled with C4.5 to achieve fast detection rates for high volume training data and bandwidth. The algorithm actively transforms discrete and nominal IPv6 packet field represented in values to nominal IDs for text based classification of the packet's data as shown in Tables 3.4 and Table 3.8.

The approach automatically provides smoothing through applying the Multinomial techniques, where field values are not available in the packet.

The core elements of the algorithms are:

1. Data Creation Tools.
2. HeuBNet6 Main.
3. HeuBNet6 Running Modes.
4. Covert Detection Engines.
5. Anomaly Detection Reporting.

3.7.1 Attack Simulation Process to Create Datasets

This is one of the critical parts of the HeuBNet6 System in compliance to RFCs which is utmost important for producing valid data for testing real world covert channel attacks. A series of testing and attack tools were used to mimic as many types of IPv6 packets as possible. These types can contain covert data or can be used to trigger a covert response for the victimised host. The suite of attack tools used for both creating training and testing data ran on a virtualized environment. The tools needed to be comprehensive enough to cover most of the selected attacks against IPv6 header, ICMPv6 header and a number of extensions headers within the scope of this project.

It was impossible to re-enact the exact scenario in which an attacker would create covert attacks on an internal system in this research. However, various

3. Proposed Detection Framework

Table 3.8: Original Nominal Values and NB formats for Selected Fields with Example Values

Header	Selected Field	Input Value	Nominal Values
Pseudo	Traffic Class	89 90	T_TC, F_TC
Pseudo	Flow Label	<i>src,port,dst,port,flow label</i> 2000::3 2001::19	T_FL F_FL
Pseudo	Hop Limit	456 457 458	HopL_INC, HopL_DEC, HopL_UNC
Pseudo	Payload Length	433 434 435	PLen_INC, PLen_DEC, PLen_UNC
Pseudo	Next Header	6 (frag) 22 (telnet)	Extra_CovHdr, No_Extra_CovHdr
Pseudo	Source Address	2001::7, 2001::13, 2001::15	Node_1, Node_2, Node_3
ICMPv6	Type & Code	1,0 255,255	Type 1_Code0 Type 255_Code255

ethical hacking attack tools are available to enable researchers to simulate attacks in controlled environments. The suggested tool and the developed tool in this thesis were used for the task of creating a novel IPv6 covert detection dataset and running live tests on the proposed system's prototype. The tools included some industry standard open source library such as IPv6 ToolKit [50]. Several other tools including customized samples were also used to streamline the process of creating specific attacks vectors which were not covered by the hacking tools aforementioned [46]. It was imperative to note that THC and the IPv6 ToolKit tools are more versed in performing intrusion and penetration tests than creating packets that contain covert attacks. This warranted further development or customisation of the currently available tools, other simple libraries or specifically written tools for covert channel attack.

Table 3.9: Selected Header Fields with Their Characteristics Values

Traffic Class (8 bits)	Flow label (20 bits)	Hop Limit (8 bits)	Payload,Length (16 bits)	Next Header (8 bits)	Source Address (128 bits)	ICMPv6 Type_Code (16 bits)	Class
F_TC	F_FL	HopL_INC	Plen_INC	No_Extra_CovHdr	Node_Unknown	Type 4_Code0	Covert
F_TC	T_FL	HopL_UNC	Plen_DEC	Extra_CovHdr	Node_1	Type 128_Code0	Covert
T_TC	T_FL	HopL_UNC	Plen_UNC	No_Extra_CovHdr	Node_2	Type 129_Code0	Normal

3.7.1.1 Create Training DataSet

Algorithm 1 - Create_Training_DataSets shows the process of creating packets to enable the system to learn how to classify covert attacks. This was from an attacker's point of view. The attacking tools were fed with the right configurations and data to create specific attacks for covert channel attacks. The actual numbers of the parameters taken into account in this stage were 8 attributes. Each attribute has various subset values from single range to multiple range type (between 1 and 10,752) giving a figure of 346,816,512 combinations of all possible instances. It was envisaged that the training data will be around 18.5 million instances from roughly 10,788 attributes. Section 3.7.3 presents the calculations of these estimates.

Algorithm 1 Create_Training_DataSet

INPUT: IPv6 Packets AttributeIDs P ,

- 1: $ATTACKTOOLS \leftarrow \{THC, Scapy, PBuchanCLib, IPv6ToolKit\}$
 - 2: **for each** attack tool $at \in ATTACKTOOLS$ **do**
 - 3: $config \leftarrow SETUP_PARAMETERS()$
 - 4: $at.CREATE_SEND_PACKETS(config)$
 - 5: $at.DISPLAY_RESPONSES(config)$
 - 6: **end for**
-

3.7.1.2 Create Testing DataSet

Algorithm 2 Create Testing DataSet

Same As Create Training DataSet with different INPUT data

Algorithm 2 - Create_Testing_DataSets mirrors the operations performed with the previous training algorithm. This will have far less testing packets to verify the correctness and performance of the system preparing for the holdout validation method. This process involves the choosing of a percentage of the afore-produced training dataset as to form the testing dataset. For instance, for the 38.18 GB training dataset produced 3.8 GB (10%) would be selected as the testing dataset. In this case a personal computer with 3.1 GHz Inter core i5 CPU 3450 and 8 GB RAM would be able to tolerate this amount of testing and training data. Almost

all settings and configurations will be identical but some values may be altered to allow for probabilistic detection during live tests. Thus, a variability in the testing data to simulate live attacks is introduced.

3.7.2 HeuBNet6 Main

This is the entry point into HeuBNet6 responsible for directing the internal processes of the system. These include starting the system in a specified mode, starting worker threads, establishing database connections, controlling and consolidating the covert detection process by routing packets to the right modules and interfacing with Netfilter Application Program Interface (API).

HeuBNet6 will be initially developed as command line program running on a Proxy Server in an internal network. Thus, all ingress and egress¹ traffic on the network passes through the server. Several instances of the program can run on the server to accomplish different tasks excluding running multiple instances in live mode. However, it is advisable to limit overhauling the server while the system is performing live packet classification. The running modes referred in this algorithm are described after this section.

¹Ingress traffic is the data originating outside the local network that is transmitted to a station within the network. Egress traffic is the data originating within the local network that is transmitted to a station outside the network.

Algorithm 3 HeuBNet6_Main

```

1: function CAPTURE_AND_PROCESS_INGRESS_PACKET(ThreadiQueue)
2:   while PData ← Receive_Network_Traffic do                                ▷ recv
3:     ThreadiQueue ← QUEUE_INGRESS_PACKET(PData)
4:     CALLBACK(PData, ThreadiQueue)
5:   end while
6: end function
7:
8: function CALLBACK(PData, ThreadQueue)
9:   P' ← FILTER_PACKET_FROM_QUEUE(PData)                                ▷ PsuedoHdr &
   ICMPv6Hdr
10:  P'' ← PREPROCESS_PACKET(P')
11:  Result ← INTELLIGENT_HEURISTIC_ALGORITHM(Mode, P'')
12:  if Mode = MODE_LIVE and ResultState ≠ BLOCKED then
13:    Result ← MNBC(ResultNBInput)
14:  end if
15:  _VERDICT(Result)
16: return r
17: end function
18:
19: function ISSUE_VERDICT(Result)
20:  if ResultClass = NORMAL or ResultState = ALLOW then
21:    ALLOW_PACKET(ResultPacketID)
22:  else
23:    DROP_PACKET(ResultPacketID, ErrorMsg)
24:    return 1
25:  end if
26:  return 0
27: end function

```

Algorithm 3 - HeuBNet6_Main shows the controller portion of the system which will be invoked during the initialisation of the system depicted by Algorithms 4, 5 and 6. Basically, each of the initialising algorithms will start an

independent process in the CPU using the function `START_NEW_THREAD(CAPTURE_AND_PROCESS_INGRESS_PACKETS, Queue_i)` informing the process to run the main controlling module function. The packet capture function is responsible for setting up network interfacing routine, receiving packets from the kernel and routing packets to and from the correct detection engine. This is done by the callback routine `CALLBACK (PData, ThreadQueue)` which receives an IPv6 packet as a series of bytes from a specified kernel queue. The packet must be preprocessed before it is sent to the Intelligent Heuristic Algorithm Engine in Algorithms 7 and 12 followed by live classification if the system is in live mode. The final task of the controller is issue a verdict whether or not to allow the packet into the network based on its classification. Normal packets will be released into the network while blocked or anomalous packets will be discarded, and an appropriate message will be sent to the sender. The message should be in the form that will stop re-transmission in the case of TCP packets or making the attacker to believe that there is no end-point if necessary.

3.7.3 HeuBNet6 Running Modes

There are three modes in which HeuBNet6 will be operated, namely *HeuBNet6_Data_Training_Mode*, *HeuBNet_MNB_Training_Mode* and *HeuBNet6_Live_Mode*.

3.7.3.1 Data Training Mode

Algorithm 4 HeuBNet6_Data_Training_Mode

INPUT: Run Mode - *Mode*, Number Of Threads - *NumThreads*

OUTPUT: NominalValue *Nom_{fl}*, Class *Class_c*.

Mode ← *MODE_CREATE_TRAINING_DATA*

NumThreads ← *NUMBER_OF_CPU_CORES*

for *i* ← 0 **to** *NumThreads* − 1 **do**

`START_NEW_THREAD(CAPTURE_AND_PROCESS_INGRESS_PACKETS, Queue_i)`

end for

Algorithm 4 - HeuBNet6_Data_Training_Mode depicts the first mode for initialising HeuBNet6 system to use all incoming packets to create training data. The *Mode* variable will be used to inform all modules the state of the system and will thus be used to guide database operations and classification tasks. For this task, the mode is set to MODE_CREATE_TRAINING_DATA and will attempt to create the 6 million training instances using the attack tools in Algorithm 1. *NumThread* must be set to the number of cores of the computer on which the HeuBNet6 System is running to enable efficient multi-threading operations.

Table 3.10: Feature Groups and Combination Estimations

	HL	PL	NH	TYCD	TC	FL	SA	DA	CLS	Total
Groups	3	3	2	10752	8	14	8	8	2	10788
Combination	1	1	1	1	1	1	1	1	1	8

- **Computation of data size estimation using feature combinations**

All fields in the IPv6 Pseudo header and ICMPv6 header in Table 3.10 can hold only one of its nominal values at a time. The minimum number of ways to select values such that one from each attribute nominal value group and the arrangement does not matter is given by the combination:

$$\prod_{i=1}^8 C_1^{n_i} \tag{3.7}$$

$$\begin{aligned} \text{Thus, } \gamma &= C_1^3 \cdot C_1^3 \cdot C_1^2 \cdot C_1^{10752} \cdot C_1^2 \cdot C_1^2 \cdot C_1^{14} \cdot C_1^2 \\ &= 3 \cdot 3 \cdot 2 \cdot 10752 \cdot 8 \cdot 8 \cdot 14 \cdot 2 \\ &= \mathbf{346,816,512} \end{aligned} \tag{3.8}$$

where 10,752 is the number of combinations of Type and Code for the 42 ICMPv6 Type numbers currently assigned by IANA [159], and γ is 346,816,512.

In order to calculate the estimated size of the training data we needed to obtain the number of characters per packet header by using the average length of all features' nominal values multiplied by the total number of features in a packet:

$$\begin{aligned}
 \text{Avg. characters per feature } \mu &= \sum_{n=1}^{10788} \text{len}(\text{Feature_Nominal_Name}) / 10788 \\
 &= 14 \\
 \text{Characters per packet, } \lambda &= \mu \cdot 8 \\
 &= 14 \times 8 \\
 &= 112 \\
 \text{Hence, estimated training data size} &= \gamma \cdot \lambda \\
 &= 346,816,512 \times 112 \\
 &= \mathbf{38,843,449,344 \text{ characters}} \approx \mathbf{36.18 \text{ GB}} \\
 &\quad (3.9)
 \end{aligned}$$

In summary, the training data is estimated to be in excess of 346,816,512 instances occupying 36.18 GB database space.

3.7.3.2 MNB Training Mode

In Algorithm 5 - HeuBNet6_MNB_Training_Mode *Mode* was set to `MODE_TRAIN_NB` to train the MNB classifier before commencing live packet classification. Unequivocally, *NumThreads* was required to be set to the number of cores the computer has as in the previous algorithm. Three more variables were introduced to capture the administrator's preference using *OvrwrMod* to overwrite the current model file being used by the classifier, a file containing the list of new nominal files *TrainFileList* with training instances shown in Figure 3.6 and the prefix of the new model files *ModFile* to be created during this training process.


```
data\nom-training-0001.dat 280869
data\nom-training-0002.dat 280865
data\nom-training-0003.dat 280867
data\nom-training-0004.dat 280868
...
data\nom-training-0033.dat 280822
```

Figure 3.6: Nominal File List for Training MNB Model

All users' input values are symbolised by a question mark (?). See Section 3.7 on conversion of nominal-values to NB format.

Algorithm 5 HeuBNet6_MNB_Training_Mode

INPUT: Run Mode - *Mode*, Number Of Threads - *NumThreads*

OUTPUT: NominalValue *Nom_{fl}*, Class *Class_c*.

Mode \leftarrow *MODE_TRAIN_NB*

NumThreads \leftarrow *NUMBER_OF_CPU_CORES*

OvrwrMod \leftarrow ?

TrainFileList \leftarrow ?

ModFile \leftarrow ?

for *i* \leftarrow 0 **to** *NumThreads* - 1 **do**

 START_NEW_THREAD(CAPTURE_AND_PROCESS_INGRESS_PACKETS, *Queue_i*)

end for

Algorithm 6 HeuBNet6_Live_Mode

INPUT: Run Mode - *Mode*, Number Of Threads - *NumThreads*

OUTPUT: NominalValue *Nom_{fl}*, Class *Class_c*.

Mode \leftarrow *MODE_CREATE_TRAINING_DATA*

NumThreads \leftarrow *NUMBER_OF_CPU_CORES*

for *i* \leftarrow 0 **to** *NumThreads* - 1 **do**

 START_NEW_THREAD(CAPTURE_AND_PROCESS_INGRESS_PACKET
 s, Queue_i)

end for

3.7.3.3 Live Mode

Algorithm 6 - HeuBNet6_Live_Mode used the model files created using Algorithm 5 and Heuristic Algorithm in passive mode. Thus, NIHA's live classification should not be taken into account since live mode is designed to perform a probabilistic classification using a probability model file as opposed to the deterministic approach NIHA uses.

For each packet received, a multinomial classification was performed to keep a count of unknown features which have been smoothed. The count of smoothed features indicated whether the training data was sufficient enough to classify new unknown attacks. If the count was beyond a preferred threshold, the classification by NIHA was used and an entry was created in the log file to allow analysis of the instance.

3.7.4 Covert Channel Detection Engines

This section presents algorithms for the proposed HeuBNet6 system and discusses the classifiers that are built using the algorithms. The two classifiers are presented as detection engines for abstraction purposes and easy functional extensibility. The HeuBNet6 Intelligent Heuristic Engine (HIHE) is presented first, followed by the HeuBNet MNBC Engine. An outline of how Decision Trees C4.5 was used to prune the training data is presented then information of generating statistical reports and analytics is presented. The section concludes by pointing out some of the issues that impeded the system's performance and accuracy.

3.7.4.1 HeuBNet6 Intelligent Heuristic Engine (HIHE)

This covert channel detection engine is the backbone of HeuBNet6 System. It is responsible for the creation of classification datasets and transforming packet's data between different formats. The engine is built using the Intelligent Heuristic Algorithm created in this research. The engine is a fully functional heuristic classifier which will provide application integration interfaces to allow different type of input sources for loading training and testing datasets. This will make the engine Operating System-independent and attract further research on the novel framework.

NIHA performed two tasks within the HeuBNet6 system. Firstly, NIHA did feature covert channel detection as shown in the functions of Algorithms 7 and 3. Each of the detection functions converted a feature's value or values within an IPv6 packet from either continuous format or string to a nominal value or set of normal values. Secondly, if the IPv6 packet must also be classified by the HeuBNet6_MNB Engine which will be discussed in Algorithm 8, NIHA would further transform the nominal values to pairs of numeric-id representation and frequency of occurrence in the packet's field such as $253:3$ or $\{620:3\ 34:1\ 16435:2\ 67343:1\}$ as expected by the engine. Having all features in the same numeric representation have improved the performance of the MNB classifier and eliminated the ambiguity of attributes. Furthermore, the size of the text to be analysed by the classifier was reduced due to the short length of the IDs. The value formats are shown with example data in Tables 3.8, 3.4 and 3.9.

Algorithm 7 HeuBNet6_Intelligent_Heuristic_Algorithm

INPUT: Run Mode - $Mode$, IPv6 Packet - $IPv6_PKT'$.

OUTPUT: NominalValue Nom_{fl} , Class $Class_c$.

- 1: $TC \leftarrow []$
- 2: $FC \leftarrow []$
- 3: $NC \leftarrow []$
- 4: $Class_c \leftarrow NULL$
- 5: $IPv6_HDR \leftarrow NULL$
- 6: $ICMPv6_HDR \leftarrow NULL$

```

7: HBH_XHDR ← NULL
8: FRAG_XHDR ← NULL
9: function INITIALISE_ENGINE(Mode, IPv6_HDR,)
10:   TC ← LOAD_TRAFFICCLASS_CORPUS(tcfile)
11:   FC ← LOAD_FLOWLABEL_CORPUS(flfile)
12:   NC ← LOAD_TOPOLOGY_CORPUS(tpfile)
13:   BL ← LOAD_BLACKLIST_CORPUS(blfile)
14:   Classc ← NORMAL
15: end function
16: function
   CHECK_IANA_RULES_RUN_ANOMALY_DETECTION(Mode, IPv6_PKT,)
17:   IPv6_HDR ← GET_IPV6_HDR(IPv6_PKT)
18:   ICMPv6_HDR ← GET_ICMPV6_HDR(IPv6_PKT)
19:   DETECT_TRAFFIC_CLASS_ANOMALY(TC, IPv6_Hdr, Classc)
20:   DETECT_FLOW_LABEL_ANOMALY(FC, IPv6_Hdr, Classc)
21:   DETECT_HOP_LIMIT_ANOMALY(FC, IPv6_HDR, Classc)
22:   DETECT_NEXT_HEADER_ANOMALY(IPv6_HDR, Classc)
23:   DETECT_PAYLOAD_LENGTH_ANOMALY(IPv6_HDR, Classc)
24:   DETECT_SOURCE_ADDRESS_ANOMALY(NC, BL, IPv6_HDR, Classc)
25:   CHECK_TYPE_CODE_PAIR(IPv6_PKT, Classc)
26: end function
27: function DETECT_TRAFFIC_CLASS_ANOMALY(TC, IPv6_Hdr, Classc)
28:   if IPv6_Hdrtc ∈ TC then{
29:     cnom ← T_TC
30:   }
31:   else{
32:     Classc ← ANOMALY
33:     cnom ← F_TC
34:   }
35:   end if
36:   return cnom, Classc
37: end function
38: function FLOW_LABEL_ANOMALY(FC, IPv6_Hdr, Classc)

```

```

39:   if  $\left( \begin{array}{l} \mathbb{FC} \not\in \text{FOUR TUPLE}(\text{IPv6\_HDR}_{src\_addr}, \text{IPv6\_HDR}_{s\_port}, \\ \text{IPv6\_HDR}_{dst\_addr}, \text{IPv6\_HDR}_{dst\_port}, \text{IPv6} \\ \text{\_HDR}_{protocol}, \text{IPv6\_HDR}_{fl}) \end{array} \right)$  then
40:      $Class_c \leftarrow ANOMALY$ 
41:      $fl_{nom} \leftarrow F\_FL$ 
42:   }
43:   else{
44:      $fl_{nom} \leftarrow T\_FL$ 
45:   }
46:   end if
47:   return  $fl_{nom}, Class_c$ 
48: end function
49: function DETECT_HOP_LIMIT_ANOMALY( $\mathbb{FC}, \text{IPv6\_HDR}, Class_c$ )
50:   if NOT_CURRENT_HOP_LIMIT_PACKET( $\text{IPv6\_HDR}_{dst\_addr}$ ) then
51:      $prevHL \leftarrow \text{STORE\_CURRENT\_HOP\_LIMIT}(\text{IPv6\_HDR}_{dst\_addr})$ 
52:   else
53:      $prevHL \leftarrow \text{STORE\_CURRENT\_HOP\_LIMIT}(\text{IPv6\_HDR}_{dst\_addr})$ 
54:   end if
55: end function
56: function DETECT_NEXT_HEADER_ANOMALY( $\text{IPv6\_HDR}, Class_c$ )
57:   if
58:      $\text{IPv6\_HDR}_{nh} \notin \text{IANA\_ASSIGNED\_INTERNET\_PROTOCOL\_NUMBERS}$  then
59:      $Class_c \leftarrow ANOMALY$ 
60:      $nh_{nom} \leftarrow \text{Extra\_CovHeader}$ 
61:   else
62:      $nh_{nom} \leftarrow \text{No\_Extra\_CovHeader}$ 
63:   end if
64:    $FrgHdrs \leftarrow \text{CHECK\_IF\_PACKET\_HAS\_FRAG\_HEADER}(\text{IPv6\_PKT})$ 
65:   if  $FrgHdrs$  then
66:      $\text{CHECK\_FRAG\_COUNT\_COVERT\_ANOMALY}(\text{IPv6\_PKT}, Class_c)$ 
67:      $\text{CHECK\_INVALID\_FRAG\_OFFSET\_COVERT\_ANOMALY}(\text{IPv6\_PKT}, Class_c)$ 
68:   end if

```

```

68:   AuthHdrs ← CHECK_IF_PACKET_HAS_AUTH_HEADER(IPv6_PKT)
69:   if AuthHdrs then
70:     CHECK_AUTH_HDR_COVERT_ANOMALY(IPv6_PKT, Classc)
71:   end if
72:   return nhnom, Classc
73: end function
74: function DETECT_PAYLOAD_LENGTH_ANOMALY(IPv6_HDR, Classc)
75:   plnom ← Plen_Match
76:   if IPv6_HDRpl ≠ CALCULATE_PACKET_LENGTH(IPv6_PKT) then
77:     Classc ← ANOMALY
78:     plnom ← Plen_Unmatch
79:   end if
80:   return plnom, Classc
81: end function
82: function
   DETECT_SOURCE_ADDRESS_ANOMALY(NC, BL, IPv6_HDR, Classc)
83:   if IPv6_HDRsa_addr ∈ NC and IPv6_HDRsa_addr ∉ BL then{
84:     sanom ← Node_nomid
85:   }
86:   else{
87:     Classc ← ANOMALY
88:     if IPv6_HDRsa_addr ∉ NC then{
89:       sanom ← NODE_NOT_FOUND
90:     }
91:     else{
92:       sanom ← Node_nomid
93:     }
94:     end if
95:   }
96:   end if
97:   return sanom, Classc
98: end function

```

The HIHE Engine is described starting from the

Check_IANA_Rules_Run_Anomaly_Detection(mode, ipv6_hdr) function which denotes the entry point into the NIHA when a packet is routed to be heuristically analysed for anomalies. NIHA performs a series of checks on the selected number of IPv6 packet fields including some extension headers and at least one upper protocol layer header if present in the packet.

The values of the eight header fields of the packet and their nominal equivalence to database include whether a covert channel was found or not, and the type of attack associated with the attempted covert channel exploit were saved to the database after all covert channel checks had been performed. The packets values were saved into the "Audit Table" while the nominal values were stored into either the training or testing tables depending on the running mode. A single database was used for simplicity at this stage.

1. Traffic Class Attribute

The detection of covert channel based on the traffic was performed by using the function `DETECT_TRAFFIC_CLASS_ANOMALY(IPv6_PKT)`. A file was used to load all valid traffic classes into the HeuBNet6 system's global Traffic Class Dictionary `TC` as shown on Line 10 - `TC ← LOAD_TRAFFICCLASS_CORPUS(tcfile)`.

- **Conversion to Nominal**

The first step in detecting a covert channel is to convert the value of the Traffic Class attribute to a nominal value. A mathematical transformation function was used, given by the vector:

$$T = \begin{bmatrix} T_{TC} \\ F_{TC} \end{bmatrix}$$

where T is the vector space φ Traffic Class nominal values.

$$f : \left\{ \begin{array}{l} \mathbb{N} \rightarrow T \\ t \mapsto f(t) \end{array} \right\} \quad (3.10)$$

where $t \in \mathbb{N}$ and $t = tc_{val}$ and $f(t) \in T$

The formula 3.10 shows that t maps to a value returned by the function $f(t)$. t is called the image of $f(t)$ and f is the mapping function. This means that for every value of t there exists a nominal value in the Traffic Class Dictionary. f associates a y (nominal value) in the codomain with each argument x (traffic class value). A corresponding nominal value is modelled for each value of \mathbb{N} in the two-value vector space T . Thus, any given value of traffic yields either the nominal value T_TC representing a True Class or F_TC representing a False Class.

$$(\exists!t \in \mathbb{N})P(f(t) \rightarrow T_{nom}) \wedge t = 101 \quad (3.11)$$

The transformation through predicate calculus using the formula 3.11 can be further described as; there exists exactly one nominal value T_{nom} calculated by f for any given traffic class value t , where t has the value of 101, which in a given corporate network could be used for provide a specified quality of service.

2. Flow Label Attribute

The flow label's covert channel detection was performed using the Flow Label Manager Subsystem which monitors the values of the incoming packet's source address, destination address, the transport protocol type and the flow label.

The file was loaded into the HeuBNet6 system's global Flow Label Dictionary \mathbb{FC} as shown in Function 9 - $\mathbb{FC} \leftarrow \text{LOAD_FLOWLABEL_CORPUS}(tcfile)$. Each line represents a connection's flow label from which a 4-tuple (source address, destination address, the transport protocol type and a flow label) is obtained. The 4-tuple structure is the extension of the 3-tuple structure recommended in RFC 6437.

Each Flow label Corpus line entry represents one flow and has the following elements:

- (a) *id*: Flow ID

- (b) *src_addr*: Source Address
- (c) *dst_addr*: Destination Address
- (d) *proto*: Protocol
- (e) *f_label*: Flow Label

Each element except *alert* is represented as follows:

- (a) Element name: Name of the feature such as *id*.
- (b) Element type: Datatype for converting the value of element for use by the Flow Manager Utility Subsystem. The datatypes will be:
 - i. i: Integer
 - ii. s: String
 - iii. v: Vector or array
- (c) Variable id: The resource id for the variable holding the element's value.
- (d) Element value: The value of the element.

The *alert* element was used to specify what elements of the flow to report when a flow label covert channel attack has been identified. The content of the *alert* element is a vector of the chosen elements separated by dollar sign (\$) as shown below:

[*alert*, *v*, 1, *\$msg1\$dst_addr\$dst_addr_port\$id*]

The detection of covert channel based on the flow was performed by the function `DETECT_FLOW_LABEL_ANOMALY(IPv6_PKT)`. The function extracted the 4-tuple structure from the incoming pack and validated it against the Flow Label Dictionary. If there was no matching found, the packet was tagged as covert ($Class_c \leftarrow ANOMALY$).

• Conversion to Nominal

The first step in detecting a covert channel was the conversion of the value of the Flow Label attribute to a nominal value. A mathematical transformation was used given by the Vector:

$id, i, 0, 1001]src_addr, s, 1, :: 3]dst_addr, s, 2, :: 12]proto, i, 1, 58]$
 $f_label, i, 2, 99]alert, v, 1, msg1]$
 \dots
 $id, i, 0, 4372]src_addr, s, 1, :: 14]dst_addr, s, 2, :: 6]proto, i, 1, 58]$
 $f_label, i, 2, 17]alert, v, 1, msg1]dst_addr, dst_addr_portid]$

Figure 3.7: Flow Label Corpus

$$\mathbb{FC} = \left[\begin{array}{l} 4 - Tuple(SrcAddr, DstAddr, Protocol, FlowLabel, ID, Alert) \\ 4 - Tuple(SrcAddr, DstAddr, Protocol, FlowLabel, ID, Alert) \\ 4 - Tuple(SrcAddr, DstAddr, Protocol, FlowLabel, ID, Alert) \\ 4 - Tuple(SrcAddr, DstAddr, Protocol, FlowLabel, ID, Alert) \\ \dots \\ 4 - Tuple(SrcAddr, DstAddr, Protocol, FlowLabel, ID, Alert) \end{array} \right]$$

where \mathbb{FC} is the vector space ζ Flow 4-Tuples Objects.

$$\mathbb{FCN} = \left[\begin{array}{l} T_FL \\ F_FL \end{array} \right]$$

where \mathbb{FCN} is the vector space φ , of Flow Label nominal values.

$$f : \left\{ \begin{array}{l} \mathbb{FC} \rightarrow \mathbb{FCN} \\ l \mapsto f(\mathbb{FC}, l) \end{array} \right\} \quad (3.12)$$

where $l \leftarrow GET_4-TUPLE(IPv6_PKT)$ and $f(\mathbb{FC}, l) \in \mathbb{FCN}$

The formula 3.12 shows that l maps to a value returned by the function $f(\mathbb{FC}, l)$. l is a 4-tuple data structure created for the Source Address, Destination Address, Protocol and Flow Label values of the incoming IPv6 packet. This means that for any combination of a 4-tuple values there must

exist a nominal value in the Flow Label Class Dictionary. f associates a y (nominal value) in the codomain with each argument x (4-tuple). Thus, for any given 4-tuple, f will yield either the nominal value T_FL representing a True Flow label or F_FL representing a False Flow Label. The values processed in equation 3.13 are taken from Table 3.8 and Figure 3.7.

$$\begin{aligned}
 & (\exists!l)P(f(l) \rightarrow L_{unom}) \wedge l \text{ is a 4-Tuple} \wedge \\
 & (l_{src_addr} = 2000 :: 3) \wedge (l_{dst_addr} = 2001 :: 7) \wedge (l_{proto} = 58) \wedge (l_{fl} = 99)
 \end{aligned}
 \tag{3.13}$$

We can further describe the transformation through predicate calculus using the formula 3.13 as there exists exactly one nominal value L_{nom} calculated by f for any given flow label 4 tuple value l . In other words, a 4-tuple with some given values such as a packet with $src_add = 2000::3$, $dst_add = 2001::7$, $proto_ = 58$ and flow label = 99 maps to a nominal value T_FL (true flow label) or F_FL (false flow label) in the vector space φ . This means that every packet can be classified based on the value of its flow label, source address and destination, consequently, it can be allowed or denied to access a particular service on the network.

3. Hop Limit Attribute

The hop limit field can be used to pass covert data by using a scheme of patterns in the data it carries. The algorithm presented detects a pattern that increases or decreases the hop limit in successive packets to provide a signaling mechanism passing bits between a pair of hosts then the packets must be tagged as covert [149].

- **Conversion to Nominal**

The first step in detecting an anomaly in the hop limit field was converting the numeric value to nominal value which clearly showed the kind of change Δ_{hl} in the hop limit value since the last received packet. The transformation is given by the vector:

$$H = \begin{bmatrix} HopL_Unc \\ HopL_Inc \\ HopL_Dec \end{bmatrix}$$

where H is the vector space φ of Hop Limit nominal values.

$$f : \left\{ \begin{array}{l} \mathbb{N} \rightarrow H \\ h \mapsto f(h) \end{array} \right\} \quad (3.14)$$

where $h \in \mathbb{N}$ and $h = HopLimit_{val}$ and $f(h) \in H$

$$(\exists! h \in \mathbb{N}) P(f(h) \rightarrow H_{nom}) \wedge h = 89 \quad (3.15)$$

The transformation can be further explained using the predicate $P(f(h) \rightarrow H_{nom})$ with a bounded variable h shown by formula 3.15. The predicate is true such that there exists exactly one nominal value for a given hop limit h value computed by f . For instance, if the transformation function $f(h)$ deems that given the hop limit value 89, there is no detected covert channel then the nominal value $HopL_{Unc}$ will be returned, otherwise either $HopL_{Inc}$ or $HopL_{Dec}$ will be returned for a detected covert channel. The packet will be tagged as covert ($Class_c \leftarrow ANOMALY$) if either $HopL_{Inc}$ or $HopL_{Dec}$ is returned.

4. Next Header Attribute

Two forms of Next Header based covert detection were suggested. First, the use of Non Protocol Number to attach extension headers with covert channels and second, Fragmentation Header exploits that maximise the use of the reserve bits in the fragmentation header.

- **Conversion to Nominal**

During each of the covert channel detection tests discussed in the section below, the value of the Next Header attribute was converted to one of the

nominal values representing the attribute in the classification stage. A mathematical transformation was used given by the vector:

$$X = \begin{bmatrix} \text{No_Extra_Covert_Header} \\ \text{Extra_Covert_Header} \end{bmatrix}$$

where X is the vector space φ of Next Header nominal values.

$$f : \left\{ \begin{array}{l} \mathbb{N} \rightarrow X \\ n \mapsto f(n) \end{array} \right\} \quad (3.16)$$

where $n \in \mathbb{N}$ and $n = nh_{val}$, and $f(n) \in X$

The transformation formula 3.16 associates a y (nominal value) in the codomain with each argument x (next header value). A corresponding nominal value was modelled for each values of \mathbb{N} existing in the two-value vector space X . Thus, any given value of the next header yields either the nominal value *No_Extra_Covert_Header* signifying that there were no anomalous extension headers detected or *Extra_Covert_Header* that at least one extension header contained a covert channel.

$$(\exists! n \in \mathbb{N}) P(f(n) \rightarrow X_{nom}) \wedge n = 58 \quad (3.17)$$

We can further describe the transformation through predicate proposition using the formula 3.17 as there exists exactly one nominal value X_{nom} calculated by f for any given next header value n and it can be 58, where n maybe an IPv6 Next Header value set by IANA.

The covert detection tests explored in this research are discussed below:

- **Next header covert detection tests**

- *Non Protocol Number Test*

If the next header value was not one of the allowed protocol numbers assigned by IANA then the packet was tagged as covert as the statement

$IPv6_HDR_{nh} \notin IANA_ASSIGNED_INTERNET_PROTOCOL_NUMBERS$ in Function $DETECT_NEXT_HEADER_ANOMALY(IPv6_HDR, Class_c)$

– *Fragmentation Exploit Test*

Two types of Fragmentation Exploit were identified, namely too many Fragments Exploit and Authentication Header Exploit.

(a) *Fragmentation: Too many Fragments Test*

The attack was detected by counting the number of fragments for a packet and checking if the packet could have originally fitted in 65,535 bytes of an IP packet or before exceeding the manually set MTU. This check is shown in the statement $CHECK_IF_PACKET_HAS_FRAG_HEADER(IPv6_PKT)$ and Functions $CHECK_FRAG_COUNT_COVERT_ANOMALY(IPv6_PKT, Class_c)$ and $CHECK_INVALID_FRAG_OFFSET_COVERT_ANOMALY(IPv6_PKT, Class_c)$. The packet was tagged as covert ($Class_c \leftarrow ANOMALY$) if the fragment was anomalous.

– *Fragmentation: Authentication Header Exploit Test*

An invalid Authentication Header can be used by an attacker to create a covert channel attack by ensuring that the host will get the covert data inside it then drop it. The detection of this covert channel is shown by statements $CHECK_IF_PACKET_HAS_AUTH_HEADER(IPv6_PKT)$ and $CHECK_AUTH_HDR_COVERT_ANOMALY(IPv6_PKT, Class_c)$. If a fragment exhibited an Authentication Header Exploit the packet was tagged as covert ($Class_c \leftarrow ANOMALY$).

5. Payload Length Attribute

If the payload length did not match the actual datagram payload after removing extra data found or the packet was not a Jumbogram but had a length of more than 65,535 bytes then the packet was tagged as covert.

- **Conversion to Nominal**

The conversion of the Payload Length attribute to nominal value can be done by the following transformation vector:

$$P = \begin{bmatrix} PLen_Unc \\ PLen_Inc \\ PLen_Dec \end{bmatrix}$$

where P is the vector space φ of Payload Length nominal values.

$$f : \left\{ \begin{array}{l} \mathbb{N} \rightarrow P \\ p \mapsto f(p) \end{array} \right\} \quad (3.18)$$

where p is the instantiation of $f(h)$. $p \in \mathbb{N}$, $p = PLen_{val}$ and $f(p) \in P$

The mapping function can also be shown as $f(p) = len$ where f calculates the size of packet and compares it against the value in the Payload Length len . If the two values are equal then there is no covert channel and the converse is true. This refers to Function 74 - DETECT_PAYLOAD_LENGTH_ANOMALY(*IPv6_HDR*, *Class_c*).

$$(\exists!p)P(f(p) \rightarrow P_{nom}) \wedge ((p \geq 1280) \wedge ((p \leq max_MTU))) \quad (3.19)$$

where 1280 *bytes* is the minimum MTU between links and *max_MTU* is the Maximum Transmission Unit of the destination node's network.

The transformation can further be described through predicate formula 3.19 as there exists exactly one nominal value P_{nom} calculated by f for any given value of payload length in the IPv6 header. If the calculated payload length matches the payload length in the header, the nominal value $PLen_Unc$ is returned signifying that there is no payload length anomaly otherwise either $PLen_Inc$ or $PLen_Dec$ is returned for the detected anomaly.

Further payload length based covert channels can be identified from Miller's work [160].

6. Source Address Attribute

An address corpus was loaded into the HeuBNet6 system's global Topology (Network) Dictionary NC for all valid and enabled addresses on the network as

shown in Function 9 - $\mathbb{FC} \leftarrow \text{LOAD_TOPOLOGY_CORPUS}(tpfile)$. If the source address was not found on the network by looking up the Topology Corpus or the source address could not be verified (pinged) then the source address was probably being used as a covert channel and was tagged as an anomaly.

- **Conversion to Nominal**

The conversion of the Source Address attribute to nominal value can be done by the following transformation vector:

$$\mathbb{NC} = \begin{bmatrix} 2001 :: 7, Node_1 \\ 2001 :: 13, Node_2 \\ 2001 :: 15, Node_3 \\ 2001 :: 19, Node_Blacklisted \\ 2000 :: 3, Node_5 \\ 2000 :: 14, Node_6 \\ \dots \\ 2000 :: 33, Node_8 \end{bmatrix}$$

where \mathbb{NC} is the vector space φ of node address - nominal ID mappings.

$$f : \left\{ \begin{array}{l} \mathbb{IP} \rightarrow \mathbb{NC} \\ s \mapsto f(s) \end{array} \right\} \quad (3.20)$$

where s is the instantiation of $f(s)$, $s \in \mathbb{IP}$ and $s = SrcAddr_{val}$, and $f(s) \in \mathbb{NC}$, and \mathbb{IP} is the vector space φ of all possible Link-Local IPv6 Addresses

The transformation formula 3.20 models that for any source address in \mathbb{IP} , a corresponding nominal value in the space \mathbb{NC} exists otherwise the nominal value $Node_Not_Found$ will be returned. Thus, any given source address will either yield the nominal value administratively assigned to it (for example,

$Node_10$) or return the nominal value $Node_Not_Found$.

$$\begin{aligned}
 (\exists s \in \mathbb{IP})P(f(s) \rightarrow S_{nom}) \vee P(Node_Not_Found) \\
 \wedge s = 2000 :: 10
 \end{aligned}
 \tag{3.21}$$

The transformation through predicate proposition using the formula 3.21 can be further described as there might be nominal value S_{nom} calculated by f for any given source address value s (2000::10) is a node in a network topology, otherwise it is inferred that a node with the address s can not be found on that network.

7. Type-Code Attribute Pair

ICMPv6 Type and Code fields are not mutually independent of each other according to the conditional independence assumption of Naïve Bayes [161]. It is easy to show that ICMPv6 code depends on the type using the tautology $ICMPv6_{code} \top ICMPv6_{type}$. Thus, ICMPv6 Code on its own is not sufficient to carry out a covert channel attack and must be used in context with an ICMPv6 Type. On the other hand, certain values of ICMPv6 Types which are independent from the value of ICMPv6 Code must be either permitted into the network or blocked as discussed in Section 3.3.

A series of detection tests were be created for every known Type-Code covert channel attack. This is shown in Function 3 - $Check_Type_Code_Pair(ThreadingQueue)$ of Algorithm 12. The Type-Code value pair was converted to a nominal representation using the concatenation process dst_return_result at the bottom of the algorithm after all covert channel tests were performed on the pair.

- **Conversion to Nominal**

The conversion of the Type and Code numeric values to a composite nominal value can be done by the following transformation vector

$$T = \begin{bmatrix} Type0_Code0 \\ Type0_Code1 \\ \dots \\ Type0_Code255 \\ Type1_Code0 \\ \dots \\ Type255_Code255 \end{bmatrix}$$

$$f : \left\{ \begin{array}{l} \mathbb{N} \rightarrow T \\ t \mapsto f(Type + ICMPv6_{type} + _ + Code + ICMPv6_{code}) \end{array} \right\} \quad (3.22)$$

where

$$ICMPv6_{type} \in \{0..4, 100, 101, 127..159, 200, 201, 255\}$$

and

$$\lim_{0 \rightarrow 255} ICMPv6_{code} \in \mathbb{N}$$

and $Type$ and $Code$ are string literals.

The function yielded nominal values belonging to the vector space φ for all values of $Type$ and $Code$ attributes. It should be noted that only the 42 ICMPv6 Types in use were analysed. Further Types can be added into the system as they get assigned by IANA but the system will require to re-train the models for the MNBC classifier.

$$\begin{aligned} & (\exists! \{t, c\} \in \mathbb{N}) P(f(t, c) \rightarrow T_{nom}) \wedge \\ & (t \geq 0) \wedge (t \leq 255) \wedge (c \geq 0) \wedge (c \leq 255) \end{aligned} \quad (3.23)$$

The transformation through predicate formula 3.23 can be further described as there exists exactly one nominal value T_{nom} calculated by f for any given value pair of type and code $\{t, c\}$ whose values are within the range 0 to 255.

- **Type and Code detection tests**

Seventeen covert channel attacks based on the received type and code values

were identified. If a covert attack was identified in a test, the packet was classified as covert with the statement $Class_c \leftarrow ANOMALY$ and the control jumped to the **DST_RETURN_RESULT** label in the algorithm. This is to stop processing Type and Code covert detection since only one of the Type-Code covert detection errors can exist for each pair of type and code values.

– **Echo Request and Reply Tests**

In the first test, it should be checked whether a packet's $ICMPv6_{type}$ is $ICMP6_ECHO_REQUEST$ (128) and $ICMPv6_{code}$ is not $ALLOWED_ECHO_REQUEST_CODE$ (0). Similarly in the second test for $ICMP6_ECHO_REPLY$ (129), it should be checked whether $ICMPv6_{code}$ is not $ALLOWED_ECHO_REPLY_CODE$ (0). If these are the case jump to **DST_RETURN_RESULT** to stop processing Type and Code covert detection.

– **Destination Unreachable Message (DUM) Test**

Tests three to seven were for ensuring that the blocked code as specified by IANA were marked to be dropped during the $ISSUE_VERDICT$ stage. If the $ICMPv6_{type}$ was $ICMP6_DST_UNREACH$ and $ICMPv6_{code}$ was in the range from 0 to 4, these are Destination Unreachable Message, Packet Too Big Message, Time Exceeded Message and Parameter Problem Message.

The DUMs were checked using the following tests:

(a) *Destination Unreachable Test 1: Route to destination does not exist*

If the $ICMPv6_{code}$ was 0, a route to the destination would be checked, for example, using a traceroute technique. The test $CHECK_ROUTE_EXISTS(ICMPv6_HDR_{dst_addr})$ tagged the packet to be dropped by setting $Result_{state} = DROP$ if a route to the destination could not be found. Program control jumped to **DST_RETURN_RESULT** to stop processing Type and Code covert detection.

(b) *Destination Unreachable Test 2: Access Denied*

If the $ICMPv6_{code}$ was 1, the destination address was checked whether it

had been administratively set not to be accessed by a firewall filter or proactively deny any access attempts. The test `CHECK_DST_ADDR_ACCESSIBLE(ICMPv6_HDR_dst_addr)` tagged the packet to be dropped by setting $Result_{state} = DROP$ if the address could not be accessed. Program control jumped to `DST_RETURN_RESULT` to stop processing Type and Code covert detection.

(c) *Destination Unreachable Test 3: Address not assigned*

If the destination address existed on the internal network by looking up the Managed Network Address Directory used by the system as shown in the test `CHECK_DST_ADDR_IN_TOPOLOGY_CORPUS(ICMPv6_HDR_dst_addr)` attempted to find if the address was active. When $ICMPv6_{code}$ was 2, the packet was tagged to be dropped by setting $Result_{state} = DROP$ if the address was not assigned by the DHCPv6 by listening to neighbour solicitation messages or tracking the addresses using DHCPv6-Stateful server. Program control jumped to `DST_RETURN_RESULT` to stop processing Type and Code covert detection.

(d) *Destination Unreachable Test 4: Address unreachable*

As in test 3, the destination address could exist on the network but it could not be reachable for some reason, for example the node was down. If the $ICMPv6_{code}$ was 3, it was checked whether the destination address was active by using a ping message or active connection tracking. The test `CHECK_DST_ADDR_REACH(ICMPv6_HDR_dst_addr)` tagged the packet to be dropped by setting $Result_{state} = DROP$ if the address could not be reached. Program control jumped to `DST_RETURN_RESULT` to stop processing Type and Code covert detection.

(e) *Destination Unreachable Test 5: Port unreachable*

The final test was when $ICMPv6_{code}$ equaled 4 for checking whether the incoming port for receiving packets was open. `CHECK_DST_ADDR_PORT_REACH(ICMPv6_HDR_dst_addr)` tagged the packet to be dropped by setting $Result_{state} = DROP$ if connection to the destination port was unreachable or prohibited. Program control

jumped to `DST_RETURN_RESULT` to stop processing Type and Code covert detection.

– *Packet Too Big Message (PTBM) Test*

The eighth test was when `ICMPv6_code` was PTBM. The size of the packet was checked by `sizeof(IPv6_PKT)` to be larger than the MTU of the destination network in test `PACKET_EXCEEDS_MTU(IPv6_PKT)`. The packet was tagged to be dropped by setting `Result_state = DROP` if its size exceeded the MTU and an entry was made in the log file for diagnostic purposes. It was envisaged that the information in the log file would be vital in configuring the right MTU for a specialised network other than just silently dropping the over-sized packets.

– *Time Exceeded Message (TEM) Test*

The ninth and tenth tests were when `ICMPv6_code` was one of TEM numbers.

(a) *Time Exceeded Message Test 1: Hop Limit Exceeded*

When `ICMPv6_code` was 0 the test `HOP_LIMIT_EXCEEDED(IPv6_PKT)` checked whether the packet's hop limit indicated that the packet had been forwarded through the routers too many times in search of its destination. In IPv6 the hop limit field is decremented when the packet passes a router. When the hop limit reaches zero, a router must drop the packet. Equally, the HeuBNet6 system dropped the packet by setting `Result_state = DROP`. This meant there was a routing loop or the initial hop limit value was too small for the packet to reach its destination.

(b) *Time Exceeded Message Test 2: Fragment reassembly time exceeded*

When `ICMPv6_code` was 1 the test `REASSEMBLY_TIME_EXCEEDED(IPv6_PKT_fhdr)` checked whether the fragmented packet's reassembly time had exceeded a certain time. The HeuBNet6 system dropped the packet by setting `Result_state = DROP` when the packet could not be reassembled in time.

– *Parameter Problem Message (PPM) Test*

The eleventh and thirteenth tests were when `ICMPv6_code` was one of PPM. Attackers can intentionally send a malformed packet to syphon data out of the network using the payload of a parameter problem message returned.

(a) *Parameter Problem Test 1: Erroneous header field*

When $ICMPv6_{code}$ was 0 the test $UNKOWN_HEADER_FIELD_FOUND(IPv6_PKT)$ checked that the fields in the packet were authentic and in a valid format. If a field was found to be erroneous the packet was dropped by setting $Result_{state} = DROP$.

(b) *Parameter Problem Test 2: Fake next header*

When $ICMPv6_{code}$ was 1 the test $UNKOWN_HEADER_TYPE(IPV6_PKT)(IPv6_PKT)$ checked that the next header field of the packet was a legitimate number as assigned by IANA. If a header type was not between 0 to 255 then the packet was dropped by setting $Result_{state} = DROP$.

(c) *Parameter Problem Test 3: Unknown option*

When $ICMPv6_{code}$ was 2 the test $UNKOWN_IPV6_OPTION(IPV6_PKT)(IPv6_PKT)$ checked that for the type of the packet received, all field names were valid. If an unknown option was found the packet was dropped by setting $Result_{state} = DROP$.

– *Multicast Listener Discovery Test*

The fourteenth test was when $ICMPv6_{code}$ was one of MLD numbers. If $ICMPv6_{type}$ was one of MLD numbers and either one or both $IPv6_PKT_{src_addr}$ and $IPv6_PKT_{dst_addr}$ were not part of the network by looking up the Managed Network Address Directory (MNAD) or link-local addresses then the packet was dropped by setting $Result_{state} = DROP$.

– *Permitted Messages Test*

On the contrary, the fifteenth test in function $ALLOW_PERMITTED_MESSAGE(IPv6_PKT)$ checked for packets carrying permitted error messages belonging to the sets UIM, EM and ETN and do not have any covert attack detected in any of the packet's fields. IANA has specified a set of ICMPv6 types which are either reserved for future use or to be allowed for experimentation and these are the types UIM, EM and ETN. UEM were not permitted in this research due to large number of Type-Code combinations (255 x 255) when UEM were not included since they are not currently in use by the IPv6 protocol. This decision

yielded only 42 x 42 Type-Code combinations for our research. A normal UIM, EM and ETN packet was allowed into the network by setting $Result_{state} = ALLOW$.

– *Neighbour Discover (ND) Test*

The sixteenth and seventh tests checked if $ICMPv6_{type}$ was Neighbour Discovery Solicitation ICMPv6 Messages (133 - 137) or Advertisement Messages (141, 142, 148, 149). For both tests, the $ICMPv6_{code}$ was checked to be zero otherwise the packet was tagged as anomalous by setting $Class_c = ANOMALY$.

– **Conversion of Nominal Values to NB Format**

According to the HeuBNet6 Live Mode, once all fields of the IPv6 packet had been checked for covert channels, NIHA further transformed the nominal values to pairs of numeric-ID representation and frequency of occurrence such as $253:3$ or $\{620:3\ 34:1\ 16435:2\ 67343:1\}$ as expected by the HeuBNet6 MNB Engine.

All known features' nominal value had their NB feature IDs recorded in a Class Map with the format:

- (a) Nominal ID: The nominal value representing a known feature (IPv6 packet field value)
- (b) NB ID: The numeric Naïve Bayes ID of the feature.
- (c) Class ID: The classification ID of the feature's value (1 means anomaly and 2 means normal)

The Class Map was loaded into the HeuBNet6 System in function `LOAD_ATTR_ID_CLASS_MAP(clsfile)`. A sample of the file is given in Figure 3.8.

The `GET_ATTRIBUTE_ID(nom_id)` function retrieved the NB ID and Class ID of a feature or token given the nominal value of the feature. All the retrieved NB IDs together with their frequencies in the packet were concatenated into a space-separated string with the eventual packet classification ID at the beginning of the string. The resultant string was passed back to HeuBNet6 Main before being passed to HeuBNet6 MNBC Engine for a probabilistic covert channel detection. The final NB formatted

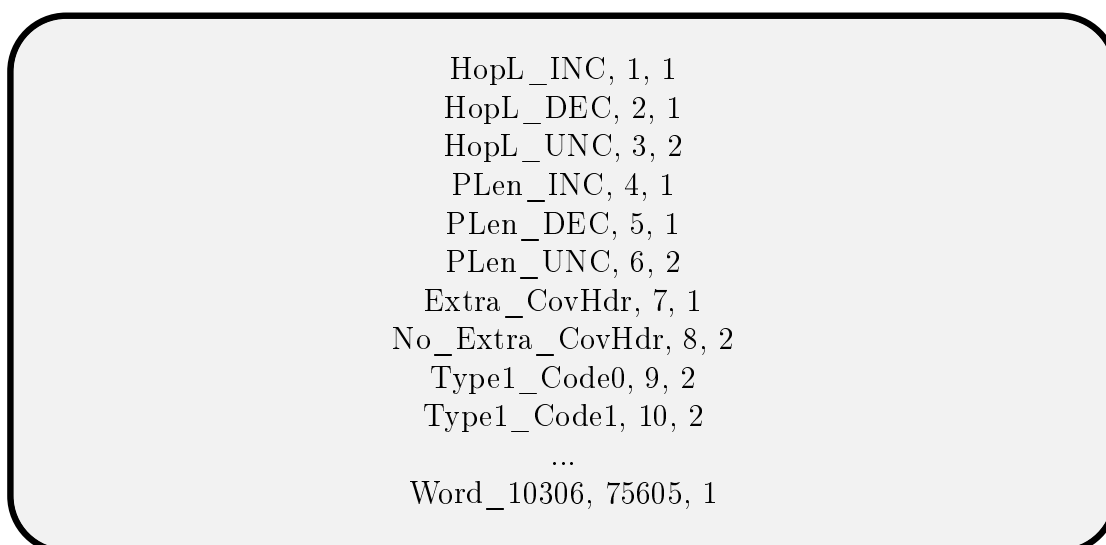


Figure 3.8: Feature Nominal-NB-Class Map

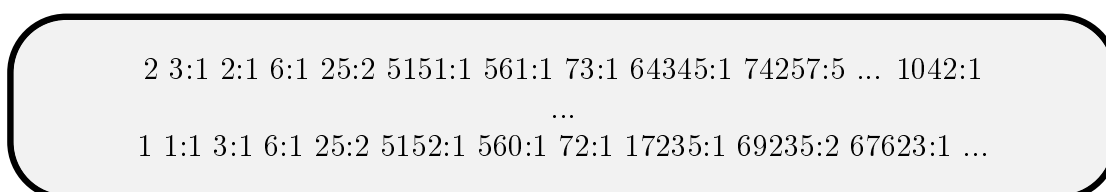


Figure 3.9: Feature Nominal-NB-Class Instances

string lookED is shown in Figure 3.9.

3.7.4.2 HeuBNet6 MNB Engine

The second covert detection engine employed text based classification to improve accuracy and performance as opposed to the Maximum A-Posteriori (MAP) probability estimate, and to improve the speed obtained by a basic Naïve Bayes classifier. This was of paramount importance in the case that most of the input features were not discrete rather interpreted by a set of rules for classifying the features' values or had nominal categories. The consequence of using an enhanced Multinomial Naïve Bayes classifier for live packet classification was the ability to achieve

almost constant time in processing packets using deep packet inspection.

In an IPv6 packet, all fields are potentially prone to covert attacks with different levels of severity, hence some features can be removed before running the classifier to concentrate on the features with the most detrimental effect. The classifier's algorithm starts with defining the class prior as follows:

$$\hat{P}(c) = \frac{N_c}{N} \quad (3.24)$$

where $\hat{P}(c)$ is the probability of the class c , N_c is the number of packets in class c and N is the total number of packets in equation 3.24 and $\hat{P}(f_k | c)$ is the conditional probability of the subset value of f given the class c in equation 3.26. $\hat{P}(c)$ and $\hat{P}(f_k | c)$ can be estimated by finding the Maximum Likelihood Estimate (MLE). MLE is simply the relative frequency and corresponds to the most likely value of each parameter such as traffic class, flow label etc., given the training data.

To find out the word count in each packet of the training data, the count of word x in all training documents belonging to Class c should be calculated. It is given by the equation summarised as:

$$\hat{P}(w | c) = \frac{\text{count}(w, c) + 1}{\text{count}(c) + |V|} \quad (3.25)$$

where $|V|$ is the size of the vocabulary (the words in the training set). The +1 in the equation is the Laplace Smoothing to avoid obtaining a conditional probability of unclassified words which results in $\frac{0}{|V|}$. Equation 3.26 shows the final probability equation which uses logarithm values of the class prior and conditional probabilities to avoid over-fitting (floating-point underflow):

$$\log \hat{P}(c) + \sum_{1 \leq k \leq n_d} \log \hat{P}(f_k | c) \quad (3.26)$$

Algorithm 9 describes the feature reduction process.

- **Multinomial Naïve Bayes Training**

raining the HeuBNet6 MNB Classifier 3 was preceded by the creation of a

model file containing the conditional probabilities of each known feature value as shown in Section 3.7. Algorithm 8 line 5 - $\mathbb{P}, \mathbb{C} \leftarrow \text{LOAD_TRAINING_DATA}(\text{modfile})$ shows the loading of the nominal-values list file created by C4.5 feature pruning into the vector of each packet's features and their frequencies in the packet \mathbb{P} and a set of class priors \mathbb{C} .

This is followed by the calculation of four variables used in the classification equations;

$F_{setsize} \leftarrow \text{MAX}(\mathbb{P}_{feat_id})$ where $F_{setsize}$ is the largest feature ID

$F_{size} \leftarrow \text{COUNT}(\mathbb{P})$ where F_{size} is the size of the set of feature IDs

$C_{setsize} \leftarrow \text{MAX}(\mathbb{C}_{class_id})$ where $C_{setsize}$ is the largest Class ID

$C_{size} \leftarrow \text{COUNT}(\mathbb{C})$ where C_{size} is the size of the set of Class IDs

$\text{PREDICTEDCLASSES} \leftarrow []$ where PREDICTEDCLASSES is the set of predicted classes, initially empty.

Lines 10 to 14 show the calculation of class priors from the set of packet's features and their frequencies \mathbb{P} and $F_{setsize}$ using the equation 3.25.

Lines 16 to 27 show calculation of feature conditional probabilities for each of the classes using the equations 3.25 and 3.26.

Lines 28 to 37 show the use of the predicted probabilities to generate a model file which would then be used for classifying packet during Live Mode.

- **MNB Packet Classification**

Multinomial Naïve Bayes' text classification is one of the best and fastest classifiers although it is simple to implement. In this research, the incoming IPv6 packet were treated as a document to be classified. The information in the packet's pseudo header, ICMPv6 header, payload or in any available extension headers and upper-layer protocol headers were converted to a uniform format as discussed earlier which the classifier. Once the trained classifier received a packet it used the model file of predicted conditional

probabilities of all available features to classify the packet as either normal or covert.

The `CLASSIFY_PACKET(packet, outfile, format)` function shows the MNB classification stage which has three inputs - the IPv6 packet, the output file to write predicted results to and the output format of the results. Line 41 - $\mathbb{P}, \mathbb{C} \leftarrow \text{EXTRACT_PACKET_FEATURES}(modfile)$ shows the extraction of features and their frequencies from the incoming packet into the sets used for the classification process followed by the calculation of feature and class ID sets variables as done during the MNB training phase from Line 42 to 47.

Line 48 starts the iteration through each feature of the packet to find their predicted conditional probabilities for each class (normal or anomaly). The process is done in three phases:

- (a) Calculate the n -arity predicted score for each of the features k from the logarithm of feature's predicted conditional probability of a given class multiplied by the frequency of the feature in the packet $n = 2$ for classes {Normal, Anomaly}

$$\log \hat{P}(c) + \sum_{1 \leq k \leq n_d} \log \hat{P}(f_k | c) \quad (3.27)$$

- (b) Calculate the n predicted classes for feature k $Class_{predicted}$ from the n -arity predicted score.

$$\text{argmax}\{c \in \mathbb{C}\}(Score_{predicted}) \quad (3.28)$$

- (c) Write the n predicted classes for feature k to the outfile in the chosen format (0 for predicted scores format and 1 for predicted probabilities format).

The final calculation is finding the accuracy of the prediction by comparing the features' test classifications against the predicted classifications expressed

as a number between 0 and 1 which is found using Iverson bracket¹ [162]:

$$1 - C(x)/s$$
$$\text{where } C(x) = \sum_{n=1}^s [feat_{ij_{testscore}} \neq feat_{ij_{predictedscore}}] \quad (3.29)$$

[...] = Iverson bracket

s = feature set size

¹In mathematics, the Iverson bracket, named after Kenneth E. Iverson, is a notation that generalises the Kronecker delta.

Algorithm 8 HeuBNet6_MNBC

INPUT: Database $DB4$, IPv6 Packets AttributeIDs P , Classification $Class_c$

OUTPUT: Prior probabilities $PRIOR$, Conditional Probabilities $CONDPRB$, Training model $Tmod$ file

```

1: procedure GET_FREQUENCY_TABLE( $a, b$ )    ▷ The g.c.d. of  $a$  and  $b$ 
2: end procedure
3: function TRAINMNBCCLASSIFIER()
4:    $modfile \leftarrow C45\_FEATURE\_PRUNING()$ 
5:    $\mathbb{P}, \mathbb{C} \leftarrow LOAD\_TRAINING\_DATA(modfile)$ 
6:    $F_{setsize} \leftarrow MAX(\mathbb{P}_{feat\_id})$ 
7:    $F_{size} \leftarrow COUNT(\mathbb{P})$ 
8:    $C_{setsize} \leftarrow MAX(\mathbb{C}_{class\_id})$ 
9:    $C_{size} \leftarrow COUNT(\mathbb{C})$ 
10:
    ▷ =====
    ▷ Calculate class prior probabilities given by:  $\hat{P}(c) = \frac{N_c}{N}$ 
    ▷ =====
11:    $PRIOR \leftarrow []$ 
12:
     $CLASSFREQTABLE \leftarrow COUNT\_EACH\_CLASS\_FREQ(\mathbb{C}, C_{setsize}, C_{size})$ 
13:   for  $i \leftarrow 0$  to  $C_{setsize} - 1$  do
14:      $PRIOR[i] \leftarrow CLASSFREQTABLE[i] / C_{size}$ 
15:   end for
16:
    ▷ =====
    ▷ Calculate feature conditional probabilities given by:
     $\hat{P}(w | c) = \frac{count(w,c)+1}{count(c)+|V|}$ 
    ▷ =====
17:    $CONDPRB \leftarrow []$ 
18:    $FEATCLASSFREQTABLE \leftarrow$ 
     $CREATE\_FEAT\_FREQ\_TABLE(\mathbb{P}, \mathbb{C}, F_{setsize}, C_{size})$ 

```

```

19:   CLASSFREQSUM                                     ←
    CALC_CLASS_FREQ_SUM(FEATCLASSFREQTABLE,  $\mathbb{P}$ ,  $C_{size}$ )
20:
21:   for  $i \leftarrow 0$  to  $F_{setsize} - 1$  do
22:     FEATPRB  $\leftarrow$  []
23:     for  $i \leftarrow 0$  to  $F_{setsize} - 1$  do
24:       FEATPRB[ $j$ ]                                     ←
         $(1 + FEATCLASSFREQTABLE[i][j]) / (F_{setsize} + CLASSFREQSUM[j])$ 
25:     end for
26:     CON DPRB[ $i$ ]  $\leftarrow$  FEATPRB
27:   end for
28:
    ▷ =====
    ▷ Save generated model to file
    ▷ =====
29:   for each class probability  $cp \in PRIOR$  do
30:      $Tmod \leftarrow$  APPEND( $cp$ )
31:   end for
32:    $Tmod \leftarrow$  APPEND_NEW_LINE()
33:   for each feature cond probabilities  $cp \in CON DPRB$  do
34:     for each probability  $p \in cp$  do
35:        $Tmod \leftarrow$  APPEND( $p$ )
36:     end for
37:      $Tmod \leftarrow$  APPEND_NEW_LINE()
38:   end for
    return  $PRIOR, CON DPRB, Tmod$ 
39: end function
40: function CLASSIFY_PACKET( $packet, out\ file, format$ )
41:    $\mathbb{P}, \mathbb{C} \leftarrow$  EXTRACT_PACKET_FEATURES( $packet$ )
42:    $F_{setsize} \leftarrow$  MAX( $\mathbb{P}_{feat\_id}$ )
43:    $F_{size} \leftarrow$  COUNT( $\mathbb{P}$ )
44:    $C_{setsize} \leftarrow$  MAX( $\mathbb{C}_{class\_id}$ )
45:    $C_{size} \leftarrow$  COUNT( $\mathbb{C}$ )

```

```

46:
47:   PREDICTEDCLASSES ← []
48:   for each feature  $f \in \mathbb{P}_{feats}$  do
49:     SCOREpredicted ← []
    ▷ =====
    ▷ Calculate predicted log prob using  $\log \hat{P}(c) + \sum_{1 \leq k \leq n_d} \log \hat{P}(f_k | c)$ 
    ▷ =====
50:     SCOREpredicted ← PREDICT_LOGPRB( $f$ )
    ▷ =====
    ▷ Calculate  $Class_{predicted}$  using  $\text{argmax}_{c \in \mathbb{C}}(SCORE_{predicted})$ 
    ▷ =====
51:     CLASSpredicted ← CLASS_FROM_SCORE( $SCORE_{predicted}$ )
52:     PREDICTEDCLASSES ← CLASSpredicted           ▷ add to array
53:
    WRITE_CLASS_TO_FILE( $outfile, Class_{predicted}, SCORE_{predicted}, format$ )
54:   end for
55:
56:   accuracy ←  $(1 - \text{COUNT}(\mathbb{C} \setminus \text{PREDICTEDCLASSES}) / C_{size})$ 
57: return accuracy
58: end function

```

3.7.4.3 C4.5 Feature Pruning

Stored training datasets created through the NIHA Algorithm were used to generate a training file for the MNB classifier. The data needed to undergo a pruning process as shown in Algorithm 9. Entries from the training database were read one at a time and converted to an array of features which were evaluated using Information Gain.

The pruning process organised the features with the most gaining information into a generated ranking table. A probability threshold was then used to reduce the redundant features. The downside of this approach when applied to IPv6 covert detection was that every feature could theoretically be used by attackers to perform attacks whose damage can be

measured by the importance of the attacked host network and information on it. If a basic form of Naïve Bayes was to be used for classifying the packets that may have had some of the eliminated features pruned, the detection would have led to many False Negatives as the least relevant features with covert channel capability ignored. Hence, C4.5 pruning was performed knowing that the Multinomial Naïve Bayes would subsequently be used.

Algorithm 9 HeuBNet6_C45_Feature_Pruning

INPUT: Database TDb , IPv6 Packets' AttributeIDs P , Classification $Class_c$

OUTPUT: Prior probabilities $PRIOR$, Conditional Probabilities $CONDPRB$, Training model $Tmod$ file

1: RUN_C45_PRUNING()

2:

3: **function** C45_FEATURE_PRUNING($TDb, nbtranfilelist$)

Equations for calculating Feature Gain Ratio:

$$InformationGain(A) = I(S_1, S_2, \dots, S_m) = - \sum_{k=1}^m \frac{S_i}{S} \log_2 \frac{S_i}{S} \quad (3.30)$$

$$SplitInformation(A) = - \sum_{j=1}^x \frac{S_j}{S} \log_2 \frac{S_j}{S} \quad (3.31)$$

$$GainRatio(A) = \frac{InformationGain(A)}{SplitInformation(A)} \quad (3.32)$$

$RankingTable \leftarrow []$

4: **for each** record $rec \in TDB_TT$ training table **do**

5: **for each** feature $feat \in rec$ **do**

6: $feat_{gr_c} \leftarrow CALCULATE_GAIN_RATIO(feat, Class_c)$

7: $RankingTable_{ij} \leftarrow APPEND(argmax(feat_{gr_c}))$

8: **end for**

9: **end for**

10: REMOVE_LOW_GAIN_FEATURES($RankingTable_{ij}$)

11: **while** !EndOfRankingTable **do**


```

12:     modfilei ← CREATE_MNB_TRAINING_FILE(RankingTableij)
13:     nbtranfilelist ← APPEND(filename(modfilei))
14:   end while
15: return nbtranfilelist
16: end function

```

- **MNB Training data size estimation**

This section shows the estimation of the number of model files created during C45 Pruning. In order to calculate the estimated size of the model training data we needed to obtain the number of characters per line by using the average length of all features nominal ID and frequency multiplied by the total number of features per line:

$$\begin{aligned}
 \text{Avg. characters per feature, } \mu &= \sum_{n=1}^{10788} \text{len}(\text{Feature_Nominal_ID}) / 10788 \\
 &= 7 \\
 \text{Characters per line, } \lambda &= \mu \cdot 8 \\
 &= 7 \times 8 \text{ features} \\
 &= 56 \\
 \text{Hence, estimated training data size} &= \gamma \cdot \lambda \\
 &= 346,816,512 \times 56 \\
 &= 19,421,724,672 \text{ characters} \approx 18.09 \text{GB} \\
 &\quad (3.33)
 \end{aligned}$$

The 18.09 GB MNB training data would be split into 1235 files of less than 30MB each with 280,869 lines to ease the importing process into the HeuBNet6 system. The file contains a list of training files looked as shown in Figures 3.6. A sample of the training file is shown in Figure 3.10.

```

id,i,0,1001]src_addr,s,1,::3]dst_addr,s,2,::12]proto,i,1,58]
  f_label,i,2,99]alert,v,1,msg1]
  ...
id,i,0,4372]src_addr,s,1,::14]dst_addr,s,2,::6]proto,i,1,58]
  f_label,i,2,17]alert,v,1,msg1dst_addrdst_addr_portid]

```

Figure 3.10: Flow Label Corpus

3.7.5 Anomaly Detection Reporting and Analysis

This section covers the production of various reports for analysing the effect of features on overall covert detection, setting of reporting intervals and how the HeuBNet6 would be validated using WEKA. A log file to be created for alerts is presented then various statistical reports and analytics are suggested.

3.7.5.1 Alarm

This section covers the reporting of anomalies for administrators to ensure that new suspicious threats and false detections were carefully analysed and to allow fine tuning of the training datasets. A report of detection of anomalies in a form of a log file containing selected details of each anomalous packet such as IPv6 Packet ID, Date and Time, Source Address and Port, Destination Address and Port, Protocol, Attack Type, Severity was produced at a fixed or chosen time interval. The log was designed to be checked by an administrator to ascertain the severity of specific attacks and to identify attackers and compromised or targeted hosts. Algorithm 10 shows the process of writing the details of a detected anomalous packet.

Algorithm 10 HeuBNet6_Alarm

- 1: When a message is received from the HeuBNet6 Main (the system controller);
 - 2: Create a log entry from the anomalous packet with the following details; IPv6 Packet ID, Date and Time, Source Address and Port, Destination Address and Port, Protocol, Attack Type,
 - 3: Write log entry to file.
 - 4: Check if the set prompt time has expired to alert an administrator to review the detected anomalies
-

3.7.5.2 Statistical Reporting and Analytics

This section presents a series of forms proposed for reporting detected anomalies and the production of statistical and analytical reports for checking accuracy and system performance. Statistical reporting may allow a fine grained analysis of the various attributes and their effect on performance and accuracy, while analytical reports would provide a coarse grained assessment of the system elements with recommendations.

(a) Statistical Report

The statistical report will include the following items:

- i. Attribute score predictive analysis.
- ii. Attribute probability analysis.
- iii. Detection Rates (DR) for True Positive Rate (TPR), False Positive Rate (FPR), True Negative Rate (TNR).
- iv. Classification accuracy for MNB and NIHA.
- v. Liable Covert Attacks in features comparison and analysis.
- vi. Statistical Figures.

(b) Analytical Report

Analytical reports may provide the means of assessing various aspects of the proposed system for proof of correctness, performance assessment and recommendations for future work. The reports would include the following:

- i. Behaviour of the NIHA in comparison to Finite Automata.

- ii. Performance of MNBA in comparison to NBA.
- iii. Time elapsed in detection and accuracy rate for NIHA and MNBA.
- iv. True Positive Rate (TPR), False Positive RATE (FPR), True Negative Rate (TNR), Detection Rate (DR) analysis.
- v. Confusion Matrix Figures.

3.7.6 Exporting Cross Validation ARFF Files

WEKA was used to perform cross validation process and certain methodologies will be highlighted and used for this objective such as hold out method. Section 5.1 discusses all cross validation methods. There is to date no similar IPv6 Dataset or covert channel detection system available. The two closest researches in IPv6 security are Bro and Suricata [163], although they are more focused on Intrusion Detection Systems. The benchmark training the testing data from HeuBNet6 database was exported to two ARFF format files that were to be loaded into WEKA. Thereafter, a series of algorithms such as Genetic and Fuzzy, C45, Naïve Bayes and Multinomial Naïve Bayes were run. Comparative analysis between HeuBNet6 and WEKA results, FP, FN, error rates, performance runtime and various other indicators are discussed in Chapter 5.

Algorithm 11 shows the creation of WEKA training and testing data.

Algorithm 11 HeuBNet6_Export_ARRF_Files

INPUT: Database *DB*

OUTPUT: WEKA Training File *NIHATrain.ARRF*,

WEKA Testing File *NIHATest.ARRF*

1: **WEKA TRAINING ARRF FILE:**

2: Create a new ARRF file named *NIHATrain.ARRF* on the File System

3: Create ARRF Headers as follows;

4: *@relation 'NIHATest'*

5: *@attribute 'class' {'1', '2'}*

6: *@attribute 'Packet' string*

7: *@data*

3. Proposed Detection Framework

```
8: Connect to Testing Database using a Database Manager Helper, DBMAN
9: Read all tuples from Training Data Table
10: for each  $t$  in  $r$  do
11:   Serialise tuple's nominal values (except  $tuple_{id}$  and  $date$ ) and append
   line to output ARRF File
12: end for
13: WEKA TESTING ARRF FILE:
14: ARRF File named NIHATest.ARRF with ARRF headers as shown on
   line 3 to 7
15: Read a subset of tuples (approx. N instances) from Testing Data Table
   using DBMAN
16: for  $tupleRow \leftarrow 0$  to  $NumOfTuples - 1$  do
17:   Serialise tuple's nominal values (except  $tuple_{id}$  and  $date$  and append line
   to output ARRF File
18: end for
```

3.8 Summary

In this chapter, a novel HeuBNet6 system is presented to detect and classify covert channels. This security system uses a new Network Intelligent Heuristic Algorithm (NIHA) as a primary prevention and detection module. Then a secondary security technique in the classification module is presented to tackle and classify covert channels statistically in IPv6. NIHA creates the essential part of the process which is the primary data as well as the anomaly detection part. The covert channel instances created in this process are around 18.5 millions instances from 10,788 attributes. A vast number of attacks based on this large feature set can be performed against security systems belonging to legitimate governments, agencies and private sectors as primary targets by attackers. The developed datasets are proposed for IPv6 malware and covert channels detection.

This chapter summarizes four main contributions: a new multi-thread system to deal with high speed of network stream, a novel Network Intelligent Heuristic Algorithm (NIHA), new primary data containing new IPv6 covert channel instances in a new different format, and an application of Multinomial Naïve Bayes classifier is suggested as a new shared-knowledge security system.

Existed approaches are different from and less effective than this new approach. In the pre-processing of data, network simulation data has been transformed into human readable data to ease and facilitate the transportation, classification and detection processes without causing packet loss which affected vital mechanisms of the IPv6 such as error messaging, experimentation messages and new IANA assigned numbers. In this approach, more than 25 types of possible covert channels attacks were presented in different modes. These channels were tested in four different environments LAN design; two of them had possible connection to the internet and two had not. Each network design gave a feasible outcome as shown in 4.5.

In this chapter, a novel approach was suggested as a hybrid security framework to tackle the sophisticated covert channel security issues in the packet headers of the incomplete network protocol IPv6.

Chapter 4

Experimental Methodology

4.1 Introduction

In Chapter 3, a new framework to create a security system (HeuBNet6) for covert channel detection and classification was presented. It operates as an advanced Intrusion Prevention System (IPS). The system is based on a few modules to specify attack behaviour (normal or abnormal) with Intelligent Heuristic algorithms. The implementation of the system identified data types received from filtered packets, categorized the instances, and transformed them into specific attributes with their subset values. These values were passed into data pruning process (purification) to remove the unwanted, repeated and unimportant subset values within the instances. For this purpose, Multinomial Naïve Bayes algorithm (MNB) prior to the issue verdict process was modelled, and issue verdict main function was used either to accept or reject the packets. Raising an alarm for covert and unknown attack detection, this was vital at a set time interval to check the log. Independently, this framework was developed to create a multi-layered and multi-threaded security detection which did a process of verification on the covert channel instances. The outcome of this development consisted of two folds. The first crucial part was saving the training and testing data into an SQL database which was used to produce two qualitative reports, namely statistical and analytical. The second part consisted of a complete report in final version format including all labelled data to be used as an input into

cross validation method.

This chapter aims to present the experimental methodology which demonstrates the capabilities of the suggested system to detect misuse based attacks. This chapter is organised as follows: Section 4.2 explains the experimental design topology set up. Section 4.3 describes the testing environment. Section 4.4 explains the main and sub attack scenarios. Section 4.5 describes the implementation of HeuBNet6 system components. Finally, section 4.6 produces a summary of this chapter.

Table 4.1: Firewall static wall

Protocol	Network Address	Gateway
IPv6	2001::1/64	2000::1/64
IPv6	::1/0	2001::3
IPv4	0.0.0.0/0	192.168.20.1

4.2 Network Topology

The experiment network architecture was designed according to the attack scenarios which were planned to test the system and hypothesis. This architecture design was drawn effectively for the implementation process. Figure 4.1 shows the components of the testbed network architecture which was used in this research. Graphical Network Simulator 3 (GNS3) [164] version GNS3-1.5.1-all-in-one was used. It is an open source software and used for computer network emulation through connecting real and virtual computers. This method is flexible for complex network topologies [165]; moreover, it emulates Cisco IOS images using Dynamips. Table 4.2 shows the IPv6 link-local and global addresses of the tested virtual machines. All incoming and outgoing traffic should be redirected through HeuBNet6 system which works as an active warden. LAN 3 and LAN 4 can access bi-direct within the network while LAN 1 and LAN 2 have same access points within the network through the server. Table 4.3 shows the configuration details which performed to prepare the testbed

environment and Table 4.1 shows the static firewall details. The testbed network consisted of the following hardware and software:

1. **HeuBNet6 Proxy System:** is installed on Linux based operating system: ubuntu-14.04.3-desktop-amd64.iso LTS.
2. **Firewall:** Palo Alto firewall was configured into the architecture.
3. **Attackers:** three attackers were represented by three machines running Linux OS ubuntu-14.04.3-desktop-amd64.iso TLS (the same version OS for all three attacker).
4. **Victims:** four victims were set up using Linux OS ubuntu-14.04.3-desktop-amd64.iso TLS (the same version for all four victims).
5. **Routers:** two CISCO routers were installed virtually, they are:
 - c7200 with c7200-advipservicesk9-mz.122-33.SRE6.image
 - c3725-adventerprisek9-mz124-11.image.
6. **Switches:** four switches were used.

4.3 Setting Up Testing Environment

The system was implemented in C++ programming language on a personal computer with 3.1 GHz Intel Core i5 CPU 3450 and 8 GB RAM. The testing environment was suggested to be in a controlled network lab environment rather than to run it live on the university or any public network. This is due to the university compliance with network security policy according to data protection Act 1998 [166]. Table 4.2 shows the GNS3 testbed devices and its details.

4.3.1 Virtual Machine Environment

Oracle virtual machine box with Graphical User Interface (GUI) Version 5.0.16 r105871 was used as the experimental environment. Through these sessions, results can be observed about how deep packet inspection is performed. For the suggested simulation topology design, 4 LANs were created as shown in Figure 4.1.

Table 4.2: Testbed devices IPv6 link-local/global addresses with MAC

Device Name/In GNS3	Interface	Static IP	Link-local A.	VLAN	OS
HeuB1 (Stockholm)	Eth0, Eth1	2001::1,2000::1	Not necessary	IDPS Server	Linux (Ubuntu)
Attacker 1 (Dexter)	Eth0	2000::3	fe80::a00:27ff:fe1f:54ca	1	Ubuntu
Victim 2 (Victim2)	Eth0	2000::17	fe80::a00:27ff:efd:c:f31	1	TinyCore
PC2 (server)	Eth0	2000::14	fe80::a00:27ff:fe3f:56ca	1	TinyCore
Victim 1 (Victim1)	Eth0	2001::7	fe80::a00:27ff:fe98:7b93	2	Ubuntu
Victim 3 (Victim 3)	Eth0	2001::13	fe80::a00:27ff:fe2d:55ca	2	TinyCore
Attacker 2 (Attacker 2)	Eth0	2001::15	fe80::a00:27ff:fe30:499f	2	Ubuntu

Table 4.3: HeuBNet6 experiment configuration check list

Function	Configuration Function	Ubuntu Proxy Server	Ubuntu Proxy Server	Ubuntu Proxy Server	Palo Alto PA 500
Testing	Ping from in Server to Firewall VLAN2001	✓	✓	✓	✓
Testing	Ping from LAN 1 Attacker 1 to out HeuBNet6 2000	✓	✓	✓	✓
Testing	Ping from LAN2 Attacker 2 to out HeuBNet6 2001	✓	✓	✓	✓
Testing	Ping from out Victim1 to Attacker1 2000	✓	✓	✓	✓
IPv6	All IPv6 Addresses	✓	✓	✓	✓
IPv6	Static IPv6 Routes	✓	✓	✓	✓
IPv6	Router Advertisements (with O flag if DHCP V6)	✓	✓	✓	✓
IPv6	Stateless DHCPv6 Server for Assigning DNS	✗	✗	✓	✗
IPv6	Stateful DHCPv6 Server (DHCPv6 Tests only)	✗	✗	✗	✗
Config	DNS Server (10.10.10.10)	✓	✓	✓	✓
Config	NTP Server (UK.pool.ntu.org)	✗	✗	✓	✓
Config	Network Policies IPv6	✓	✓	✓	✓

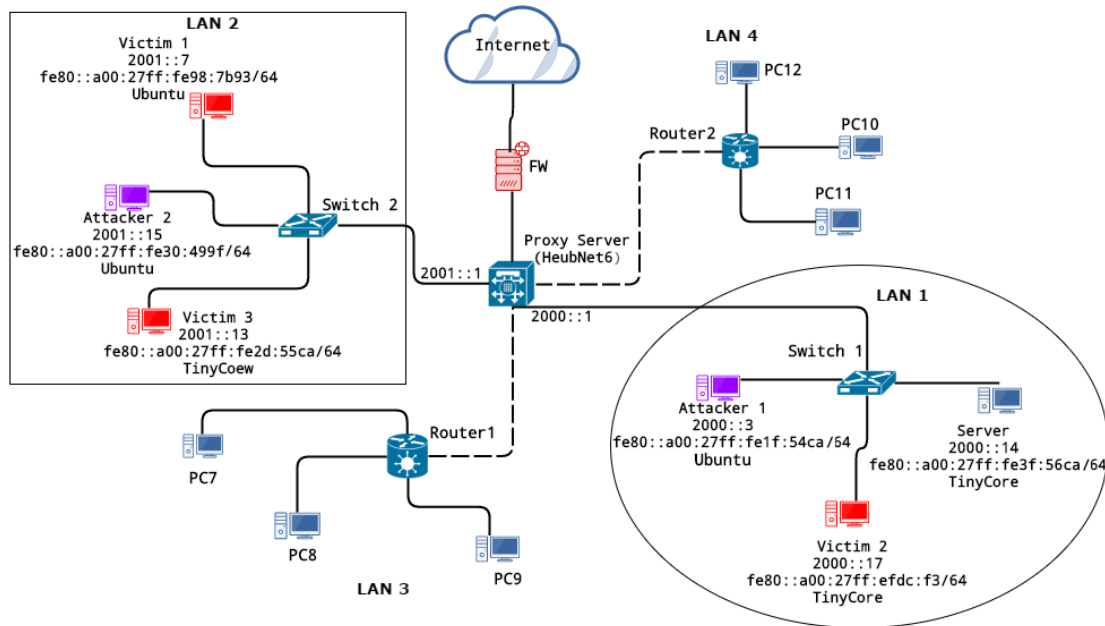


Figure 4.1: Suggested LAN Topology

4.3.2 Covert Channel Attack Tool

There are few security assessment and analysis tools for IPv6 network. Not so many tools are available for performing simulation and emulation as well. One of the most popular attack simulation tool is The Hackers' Choice¹ (THC-IPv6 Toolkit) [50]. However this tool does not provide extensibility in needed functionalities during the research work. Moreover, it lacks some of the deep packet inspection analysis mechanism which limits the testing range. Hence, a security assessment and attack simulation tool was created to perform some attacks against legitimate targets in the design topology of the thesis.

Iptables is a generic table structure firewall. Each rule within an IP table consists of a number of classifiers which match the Iptables and one connected action is called Iptables target. Furthermore, it provides the means to categorise packets into a set of distinct sets of rules, namely:

1. Filter Tables: for making decision to allow or drop packets.

¹More information about this tool is available on-line <http://tools.kali.org/information-gathering/thc-ipv6>

2. NAT Tables: for network address translation and modifying packets before forwarding to destination.
3. Mangle Tables: for packet manipulating field values and marking for further processing.
4. Raw Tables: for connection tracking and interfacing with the Netfilter framework.
5. Security Tables: for per-packet or per-connection setting of internal SELinux security context marks on packets.

On the other hand, Netfilter consists of a set of hooks as follows:

- `NF_IP_PRE_ROUTING`, `NF_IP_LOCAL_IN`, `NF_IP_FORWARD`,
- `NF_IP_LOCAL_OUT`
- `NF_IP_POST_ROUTING` hooks

Netfilter mechanism works upon some specific rules in Linux operating systems. For example, once a packet arrives, Linux kernel extension modules register callback functions with the network stack. A callback function will be called back for every packet that traverses the respective hook within the network stack. Iptables rules redirect all traffic to a processing application which decides how to handle the packets. In the proposed HeuBNet6 system, using the Libnetfilter API will aid to deal with the high streaming speed of data up to 40 Gbits per second.

4.4 Attack Scenarios

HeuBNet6 was suggested to detect and test the investigated covert channels in IPv6 in this project. This tool can be used by all parties in different scenarios such as attacker, receiver, or sender of the overt communication. The main assumption was that Bob and Alice would be either sender, receiver (or attacker and victim) while Wendy was the warden who metaphorically represented the security system which monitored and captures messages to analyse and mitigate any attacks. Some of the simulated attacks depend on certain scenarios as explained below.

4.4.1 Scenario One

In this scenario, Alice (Attacker1) was the sender and Bob (Victim1) was the receiver of the message which was actually an attack, over the traffic. This scenario depended on the active communication since both were connecting to each other in two different VLANs normal network traffic, and free from any suspicious move.

4.4.2 Scenario Two

In this scenario, Alice prepared an attack against Bob, who was vulnerable, and sent both normal and covert ICMPv6 packets to Bob (receiver). Wendy was the warden catching, monitoring all messages, mitigating, analysing and producing the outcome reports, then blocking the attack instance after updating the database. All packets with covert data were automatically dropped before reaching their destination while all normal packets were forwarded to their destinations. All 37 test cases depended on the same main scenario mentioned in section 3.2.

4.5 HeuBNet6 System Modules

This section shows some the experimented implementation of the covert channel detection functions within the HIHE.

4.5.1 Training DataSet Creation

To create test data, a few commands will be run to simulate attacks against a few victims on the network. In the first step, iptables should be configured to put packets in two queues, route them to HeuBNet6 for classification and save the results to database. The following commands were taken from attack tools THC and IPv6ToolKit:

```
/sudo iptables -I FORWARD -i eth1 -o eth0 -p ipv6-icmp -j  
NFQUEUE
```

```
myntu@heubnet6_VirtualBox:/home /heubnet6 /libnetfilter_queue
_1.0.2 sh ./hn6-train-db.sh
```

Figure 4.2 shows a packet (blue dotted line) sent by a node (2001::15) containing various values for a destination node (2000::14) at the left. HeuBNet6 acted as a router or proxy server between the two nodes, in addition to checking the packets for covert exploits. HeuBNet6 applied classification to each packet and saved the data into the training database, which was then used for training the MNBC.

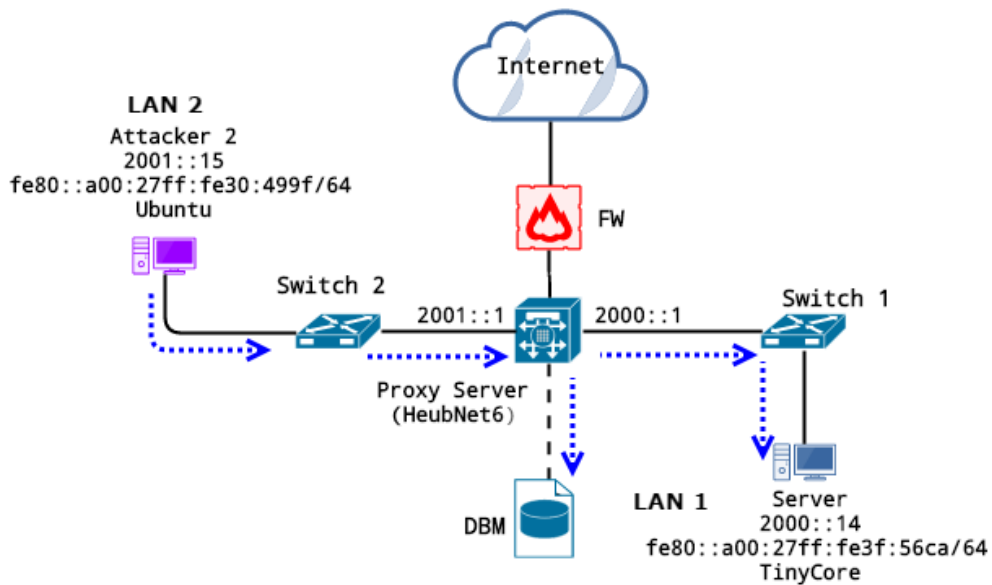


Figure 4.2: Suggested topology for training dataset creating

4.5.2 Testing Dataset Creation

To create test data, a few commands to simulate attacks against a few victims on the network were run as shown in Figure 4.1. Some of the used commands during simulation were taken from the THC, IPv6ToolKit and Scapy attack tools. The following steps were taken to prepare the testbed environment:

1. Run the file `./sendhbhfrag` created in C language to simulate the attacks on victim 2:

- `- gcc -std=c++0x -pthread -Wall -I ../../include -o sendhbhfrag tcp6_hop_frag.c`

2. Link to MySQL to store packet data into the database:

- `-/usr/include/mysql -lmysqlclient -I /usr/include/mysql`

4.5.3 Testing Heuristic Intelligent Engine

4.5.3.1 Experimental Traffic Class Anomaly Covert Detection

It was assumed in this research that all valid traffic classes on the network would have been manually created and assigned to different host's processes before any experimentation was commenced. According to the IPv6 protocol, a source node must correctly use specified traffic classes in order to access certain destination nodes. The traffic class is generally attached to an upper-layer protocol. However, in order to generate all types of packets for testing purposes, traffic class was allowed to be repeatedly used.

According to the proposed model, Table 4.4 contains the Traffic Class types in IPv6 which were allowed between nodes on the network for experimentation purposes.

Table 4.4: Traffic Class Corpus

1	23	53	78	34	122	120	220
---	----	----	----	----	-----	-----	-----

Based on Figure 4.3, two test cases of traffic class anomaly detection shown in Table 4.5 were experimented using source node Server-LAN1 and destination node Attacker2-LAN2. In the first test case, Alice sends a hidden message to Bob in the traffic class field with the value of 66. This packet is expected to be flagged as covert since traffic class is invalid. Conversely, the second test case shows a packet without a hidden message in the traffic class field with an allowed value of 23 and it is expected to be classified as normal. The actual results show that both NIHA and MNBC classifiers have verified the packet as covert and normal respectively. A brief example of how MBNC computes the classification is shown in Subsection 4.5.4.

Table 4.5: Detection of covert traffic class result using NIHA and MNBC

Src (Alice)	LAN	Dst (Bob)	LAN	Value	Expected	NIHA	MNBC
Attacker 2	2	Server	1	66	Normal	Normal	Normal
Attacker 2	2	Server	1	23	Covert	Normal	Normal

The execution commands for this attack is:

```
Alice/scapy# python testpkt.py -t 128 -c 0 - -nh 58 - -tf 23
```

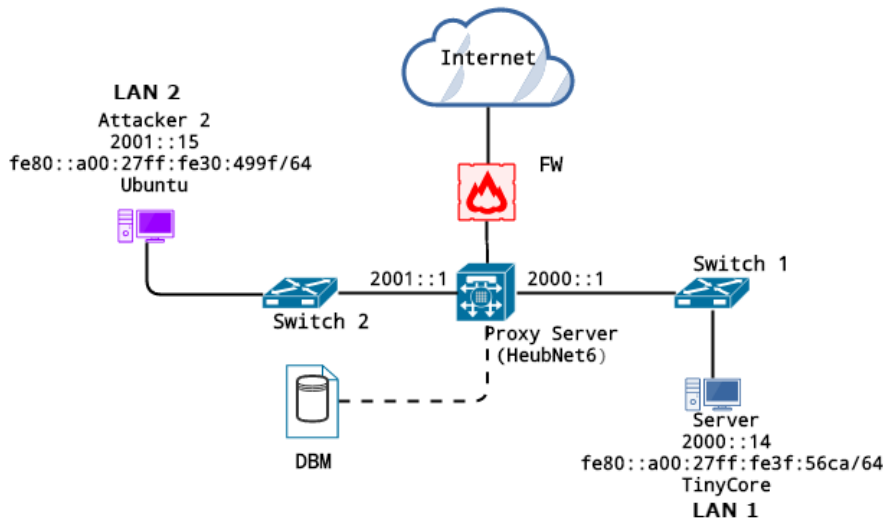


Figure 4.3: Traffic Class anomaly detection topology

4.5.3.2 Experimental Flow Label Anomaly Covert Detection

The flow label attribute has caused many concerns over the years about its true purpose and application. The attribute's specification is ambiguous in its usage and applicability due to the fact that it is unprotected during transmission and can be changed en route. Currently, no Internet-wide mechanism can depend on the attribute without applying a "best effort" quality reliability check [167]. In 2011, the flow label attribute was proposed in the RFC 7424 [168] for optimising and scaling networks using Link Aggregation Groups (LAGs) and Equal-Cost Multi-paths (ECMPs) which still remains informational.

Several implementations of the flow label have been proposed mostly based on the obsolete RFC 3697 [169, 170]. The 3-tuple (source address, destination address and the transport protocol type) flow label instead of a 5-tuple (source address, destination address, source port, destination port, and the transport protocol type) implementation is favoured in RFC 6437.

In this test, the flow label attribute was used to provide a new functionality called Network Routing Service Mechanism (NRSM) which helped to distribute traffic on the network without inspecting the packet's payload. Nodes have to use specific flow labels to access services provided by other nodes. A Flow Manager Utility Subsystem (FMUS) was created to handle the creation and administration of flow labels used on an internal network. This step was taken instead of letting a source to create a pseudo-random flow label on the fly. Flow label corpus Table 1 was created as shown in Appendix A.3.

Three test cases of flow label anomaly detection shown in Table 4.6 based on Figure 4.4 were experimented using the source node Attacker1-LAN1 and the destination node Victim1-LAN2.

- In the first test case, Alice sends a hidden message to Bob using flow label field with the hexadecimal value of 0x1BC. This packet is expected to be flagged as covert since flow label is invalid.
- The second test case shows a packet without hidden message in the same field carrying an allowed value of 0x63 and is expected to be classified as normal.
- In the third case, Alice used a different protocol value 0x16 embedding the hidden data in. This is covert. The actual results show that both NIHA and MNBC classifiers have verified the packet as covert and normal respectively.

The execution commands for this attack is:

```
/Alice/scapy/sendpkt - -sa 2001:17 -da 2000::13 -pr 58 -fl 444  
-t 128 -c 0
```

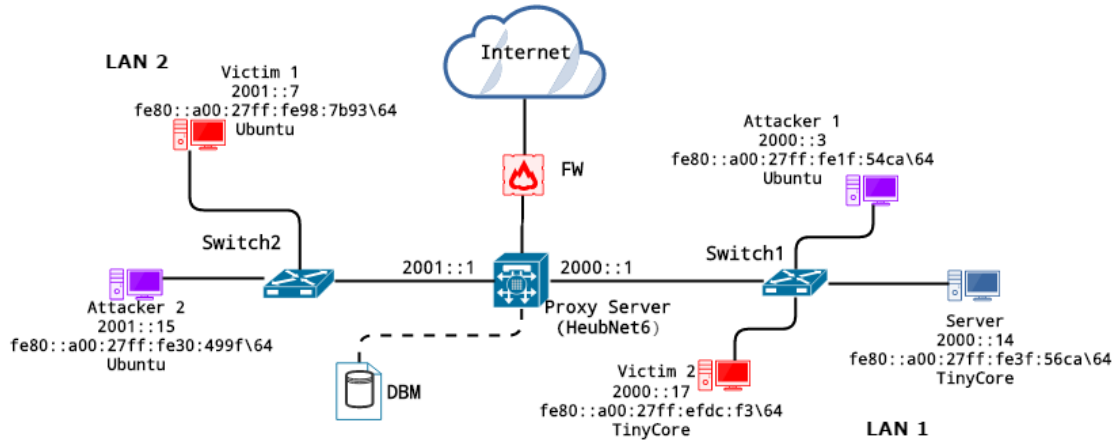


Figure 4.4: Flow label anomaly detection topology

Table 4.6: Covert detection of flow label using NIHA and MNBC

Src (Alice)	LAN	Dst (Bob)	LAN	Value	Expected	NIHA	MNBC
Attacker 1	1	Victim 1	2	444	Covert	Covert	Covert
Attacker 2	2	Server	1	99	Normal	Normal	Normal
Attacker 2	2	Victim 2	1	Proto = 22	Covert	Covert	Covert

4.5.3.3 Experimental Hop Limit Anomaly Covert Detection

Hop limit (originally Time To Live, TTL in IPv4) is used by the source address (sender) to calculate how many nodes or routers the packet will pass through before reaching the destination. An attacker can use this field to create a covert channel by systematically manipulating values in a series of packets. If a pattern is detected showing an increase or decrease in the hop limit, this occurrence will be observed in successive packets to provide a signalling mechanism passing bits between a pair of hosts then the packets must be tagged as covert.

Three test cases of hop limit anomaly detection as shown in Table 4.7 based on Figure 4.5 were experimented using source node Attacker1-LAN1 and destination node Victim1-LAN2.

- In the first test case, Alice sends a message to Bob using previous hop limit value field with the value 0xEA. This packet is expected to be flagged as normal, since hop limit is valid using the next hop limit value of hex 0x19 within a sensible number of possible routers between the source and

destination nodes.

- The second test case shows a packet with hidden value in the hop limit field. The previous value of the hop limit was set to $0xF$ while the next hop limit value was set to 789 larger than the number of routers on the network. This packet was therefore expected to be classified as covert.
- In the third case, Alice set the hop limit to a low number of hops such that the hop limit is reduced to zero before it reaches its destination. The packet was expected to be classified as covert. The actual results show that both NIHA and MNBC classifiers also have verified the packets in the three test cases as expected.

The format of command that performs the hop limit attacks is:
`/sendpkt.py -sa 2001::7 -da 2000::14 -ty 128 -nh 58 -cd 0 -fl 23
 -pd hello -tf 1 -hl 15 -pl 5`

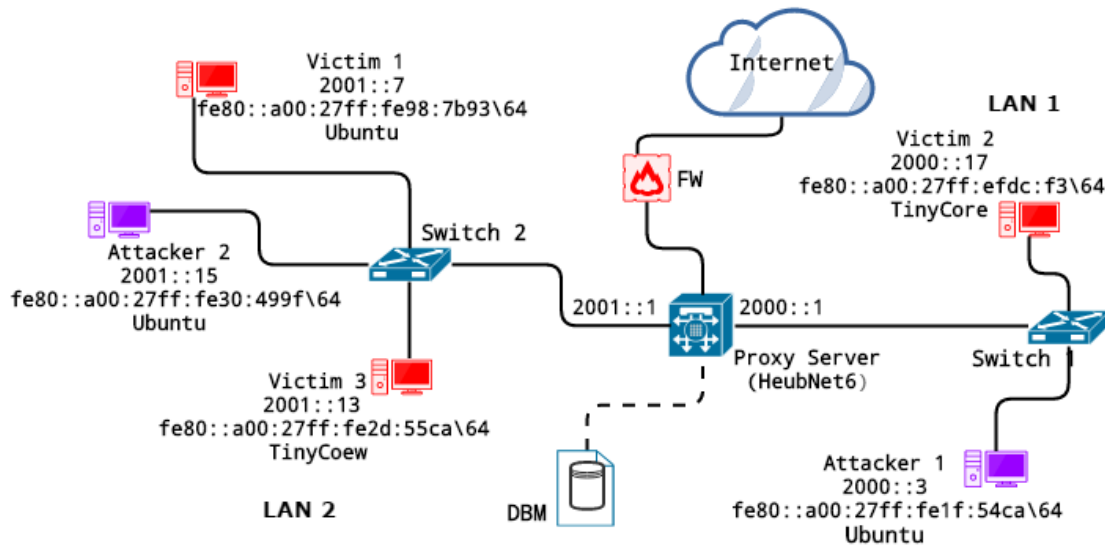


Figure 4.5: Hop Limit anomaly detection topology

Table 4.7: Covert detection of Hop Limit anomaly using NIHA and MNBC

Src (Alice)	LAN	Dst (Bob)	LAN	Value	Expected	NIHA	MNBC
Attacker 1	1	Victim 1	2	Prev=23 Next=25	Normal	Normal	Normal
Attacker 2	2	Victim 2	1	Prev=15 Next=789	Covert	Covert	Covert
Attacker 1	1	Victim 3	2	0, Not Dst	Covert	Covert	Covert

4.5.3.4 Experimental Next Header Anomaly Covert Detection

The Next Header can potentially be used by attackers to create covert channel that may exploit for both infiltrating and exfiltrating information into and out of a computer network. Fake extension headers can be appended by an attack before sending or forwarding a packet to its destination. Likewise, valid IANA protocol numbers can be used to initiate a dialogue between source and destination nodes in which the destination node will exfiltrate data using a echo reply message. The following tests have been performed:

1. *Non Protocol Number Test*: An attacker can use any number from 0 to 255 for a next header value to trigger an exfiltrating covert channel with a parameter problem reply sent back by a destination node.

2. *Fragmentation Header Exploits*: It consists of the following tests:

- *Too Many Fragments Test*

An attacker can break a packet into too many fragments with the intention of maximising the 2-bit reserve bits in each fragment to perform covert channel attacks. Additionally, an attacker can purposely create an incomplete fragment so that the whole packet can be dropped by the destination node. This will trigger a parameter problem message embedded with covert data.

- *Authentication Header Exploit Test*

An Authentication Header with an invalid Integrity Check Value (ICV) can be used together with the Hop-by-Hop extension header by an attacker. This can be done in an ingress packet knowing that it will fail

IPsec integrity protection checks and will consequently be dropped by destination node. The amount of data that an attacker can transmit from a source to destination is 1,022 bytes per packet [17].

Five test cases of extension header anomaly detection as shown in Table 4.8 based on Figure 4.6 were experimented using the source node Attacker2-LAN2 and the destination node Server-LAN1.

- In the first test case, Alice sends a message to Bob using Next Header field embedding non protocol value in hexadecimal of 0x2BC. This packet is expected to be flagged as covert since extension header is invalid.
- The second testing is to calm down the situation by sending a valid value 0x3A which will be classified as normal.
- The third test case shows a packet with hidden value (denoted by XXXXX) in PadN option of the Hop by Hop extension header. The packet was expected to be classified as covert.
- In the fourth test case, Alice specifies in the jumbogram payload option. However, the packet is not a jumbogram since it is less than 0xFFFF bytes. Eventually, the packet was expected to be covert.
- In the fifth test case, Alice set the jumbogram payload option and attaches a fragmentation extension header at the same time to embed hidden data. All actual results show that both NIHA and MNBC classifiers have verified the packets covert as expected.

The format of command execution for this attack is:

```
/sendpkt.py -sa 2001::7 -da 2000::14 -ty 128 -nh 700 -cd 0 -fl 23  
-pd hello -tf 1 -hl 15 -pl 5
```

4.5.3.5 Experimental Payload Length Anomaly Covert Detection

If the payload length did not match the actual datagram payload after removing extra data found, or the packet was not a Jumbo-gram but had a length of more than 65,535 bytes, the packet would be tagged as covert.

4. Experimental Results

Table 4.8: Covert detection of Next Header anomaly using NIHA and MNBC

Src (Alice)	LAN	Dst (Bob)	LAN	Value	Expected	NIHA	MNBC
Attacker 2	2	Server	1	Non Protocol NH = 700	Covert	Covert	Covert
Attacker 2	2	Server	1	NH = 58	Normal	Normal	Normal
Attacker 1	2	Victim 1	2	PadN=XXXXX	Covert	Covert	Covert
Attacker 2	2	Victim 2	1	Jum Plen Opt Jumbo < 65535	Covert	Covert	Covert
Attacker 1	1	Victim 1	2	Jum Plen Opt Frg Ext	Covert	Covert	Covert

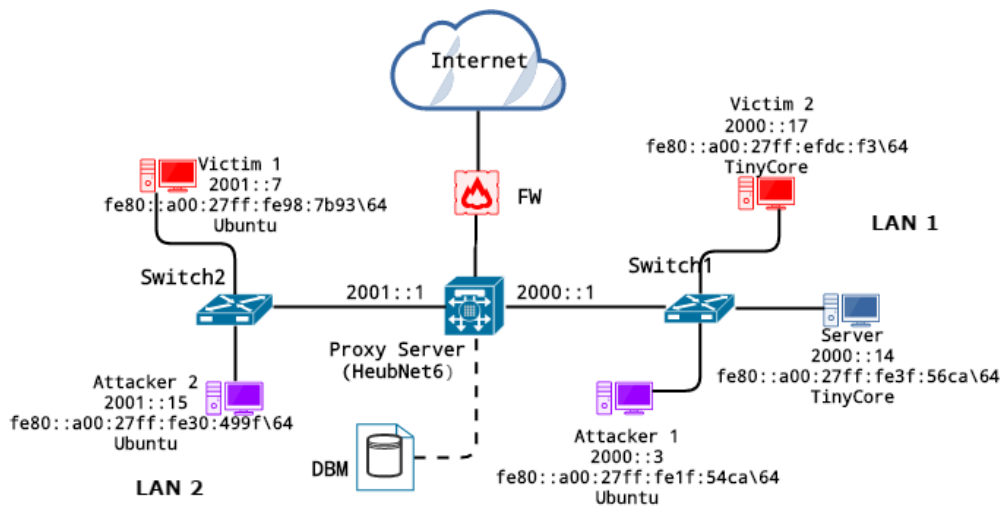


Figure 4.6: Next Header anomaly detection topology

Two test cases of payload length anomaly detection shown in Table 4.9 based on Figure 4.7 were experimented using source node Attacker1-LAN1 and destination node Victim1-LAN2. In the first test case, Alice uses the payload length field to embed a hidden message to Bob. This packet is expected to be flagged as covert since the value is invalid. Similarly, the second test case shows a packet with a larger value of the jumbo payload length within the message, which should be classified as covert. The actual results showed that both NIHA and MNBC classifiers also have verified the packet as covert and normal respectively. The execution command for this attack is:

```
myntu@abdul-VirtualBox:/sendpkt.py -sa 2001::7 -da 2000::14
-ty 128 -nh 58 -cd 0 -fl 23 -pd hello -tf 1 -hl 20 -pl 0
```

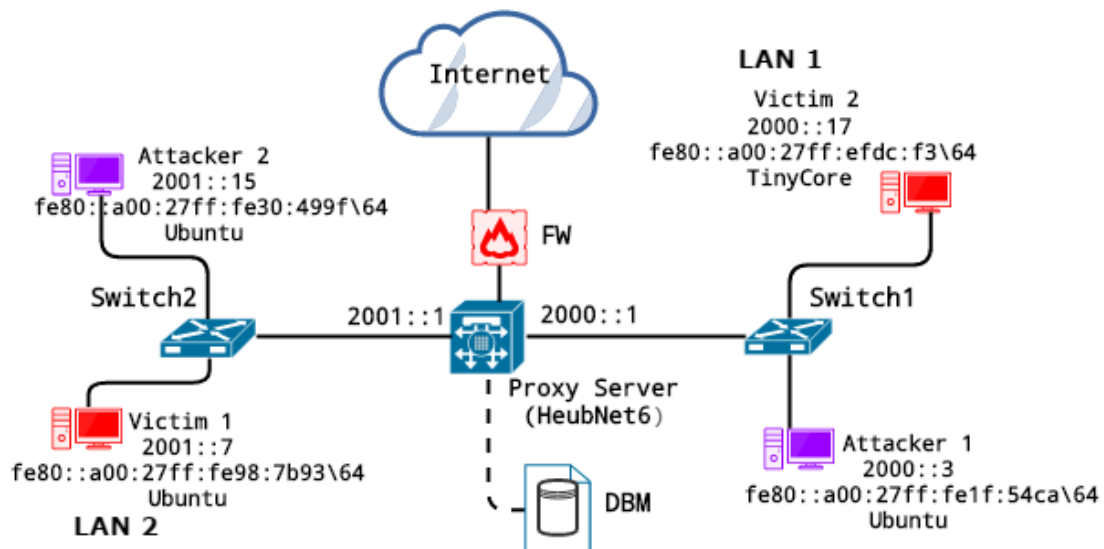


Figure 4.7: Payload Length anomaly detection topology

Further payload length based covert channels can be identified from Miller's work [160] and these are:

1. Payload length is zero, with no Hop-by-Hop options and Jumbo Payload option present.
2. Payload length is not zero with Jumbo Payload present.

3. Jumbo Payload option is present and Jumbo Payload length less than 65,535 bytes.
4. Jumbo Payload option is present with an Fragmentation Extension Header.

Table 4.9: Covert detection of Payload Length anomaly using NIHA and MNBC

Src (Alice)	LAN	Dst (Bob)	LAN	Value	Expected	NIHA	MNBC
Attacker 1	1	Victim 1	2	PL = 0 NoHBH Opt JumboOpt	Covert	Covert	Covert
Attacker 2	1	Victim 2	3	PL >0 Jumbo PyLD	Covert	Covert	Covert

4.5.3.6 Experimental Source Address Anomaly Covert Detection

The source address attribute can be used by an attacker to send 16 bytes of arbitrary data to a destination node. It is therefore imperative to check that the source address field contains a valid IPv6 address. Furthermore, in the proposed HeuBNet6 system, an address corpus of all nodes that can either contact or be contacted by other nodes was maintained. This was decided to provide an extra layer of network security.

Two test cases of source address anomaly detection shown in Table 4.10 based on Figure 4.8 were experimented using source node Attacker1-LAN1 and destination node Victim1-LAN2. In the first test case, Alice injects an original source address (2000::3) to Bob which is classified as normal. In the second test case, a fake source address (2000::88) was sent within the message covertly which was classified as covert. The actual results show that both NIHA and MNBC classifiers have verified the packet as covert and normal respectively.

The execution command for this attack is:

```
myntu@abdul-VirtualBox:/sendpkt.py -sa 2000::88 -da 2000::14
-ty 128 -nh 58 -cd 0 -fl 23 -pd hello -tf 1 -hl 20 -pl 0
```

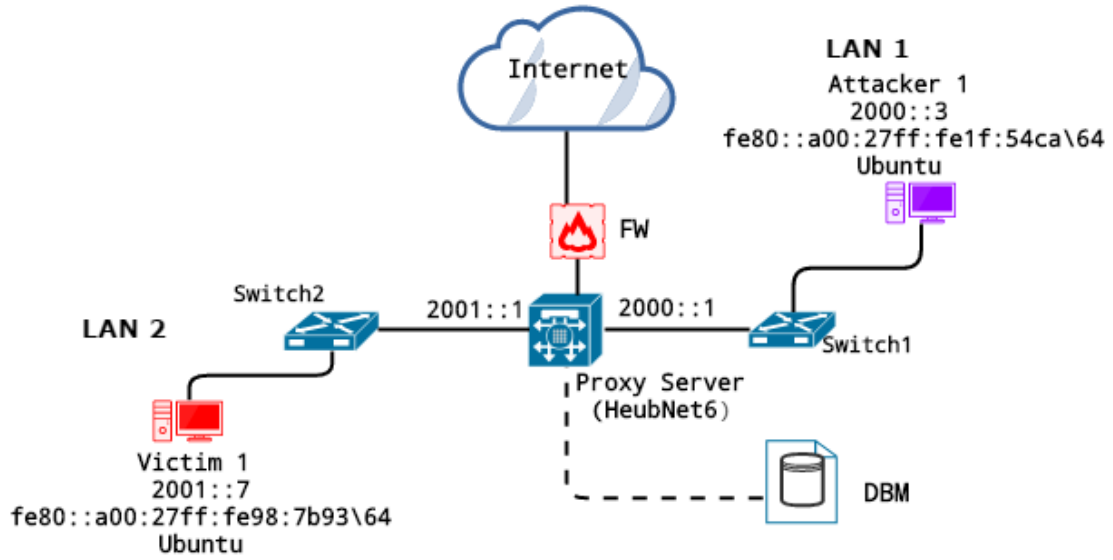


Figure 4.8: Source Address anomaly detection topology

Table 4.10: Covert detection of Fake Source Address anomaly using NIHA and MNBC

Src (Alice)	LAN	Dst (Bob)	LAN	Value	Expected	NIHA	MNBC
Attacker 1	1	Victim 1	2	2000::3	Normal	Normal	Normal
Unknown	1	Victim 1	2	2000::88	Covert	Covert	Covert

4.5.3.7 Experimental Type-Code Anomaly Detection

Some of the seventeen covert channel attacks from the algorithm were implemented and described below.

Error messages from 1-127 can be spoofed by the attacker and can be used to stop a host connecting to the destination network which results in Denial of Service (DoS). Attackers can craft their attacks knowing that the destination host will drop the packet with any of these types and return a packet with an error message. This can be performed through using either echo reply or parameter problem which can also be used to exfiltrate from the destination. In the latter error message, a huge amount of data can be extruded causing serious data breach. Since one of HeuBNet6's goals is to gather audit data for analytical and security improvement purposes, these types of packets are not dropped at the firewall-level. The packets underwent the usual covert detection process, saved to database but were not re-injected into the network regardless of the covert classification status.

Seventeen detection tests of Type-Code covert attacks were proposed in the NIHA Algorithm. For experimentation purposes the Destination Unreachable Type-Code covert attack was selected which contained the followings tests for error messages:

- *Destination Unreachable Test 1: Route to destination does not exist.*
- *Destination Unreachable Test 2: Access Denied.*
- *Destination Unreachable Test 3: Address not assigned.*
- *Destination Unreachable Test 4: Address unreachable.*
- *Destination Unreachable Test 5: Port unreachable.*

Three test cases of Type_Code anomaly detection as shown in Table 4.11 based on Figure 4.9 were experimented using source node Attacker1,2-LAN1,2 and destination node Victim1,2,3-LAN2,1,2.

- In the first test case, Alice sends a message to Bob using Type field value with the 0x80 and Code field value with 1. This packet is expected to be flagged as covert since the Type_Code sent was invalid.

4. Experimental Results

- The second test case shows an Echo Request packet with hidden value in the same field sending Type 0x80 and Code 0 expected to be classified as normal.
- In the third case, Alice sends an Echo Reply Type value 0x81 and Code 0, eventually, this packet will be normal. The actual results showed that both NIHA and MNBC classifiers also have verified the latest two cases as normal respectively.

The commands that execute these types of attacks are:

```
/scapy# python testpkt.py -sa 2001:17 -da 2000::13 -t 128 -c 1
```

```
/scapy# python testpkt.py -sa 2000::3 -da 2001::7 -t 128 -c 0
```

```
/scapy# python testpkt.py -sa 2001:15 -da 2000::17 -t 129 -c 0
```

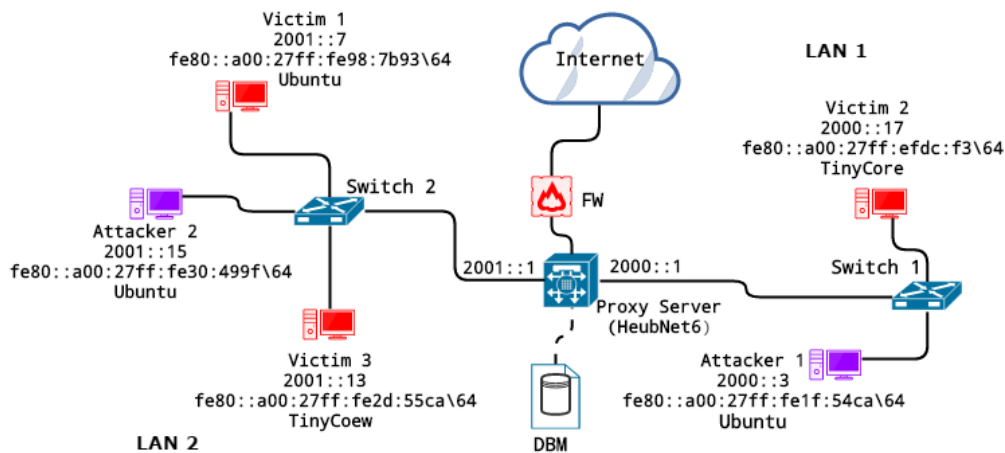


Figure 4.9: Type_Code anomaly detection topology

The Parameter Problem covert attack is associated with improper setting of field values. The intention of this action is to make the protocol to drop the packet at the destination node which in turn enables the destination node to send out hidden messages as a reply.

4. Experimental Results

Table 4.11: Covert detection of Type `_Code` using NIHA and MNBC

Src (Alice)	LAN	Dst (Bob)	LAN	Value	Expected	NIHA	MNBC
Attacker 1	1	Victim 3	2	128, 1	Covert	Covert	Covert
Attacker 2	2	Victim 2	1	128, 0	Normal	Normal	Normal
Attacker 1	1	Victim 1	2	129, 0	Covert	Normal	Normal

Parameter Problem Tests are as follows:

- *Parameter Problem Test 1: Erroneous header field.*
- *Parameter Problem Test 2: Fake next header.*
- *Parameter Problem Test 3: Unknown option.*

An experimental command that executes this type of attacks is in the form:
`/scapy# python testpkt.py -sa 2001:15 -da 2000::17 -t 299 -c 0`
 where the type value is set to a value above 255.

Two test cases were experimented. In the first instance, a packet with the Type field value of 299 was sent by Alice Attacker1-LAN1 to Bob Victim2-LAN. This packet was expected to be flagged as covert. In the second case, a packet with Type code value of 128 was sent from Alice Attacker2-LAN2 to Bob Server-LAN1. NIHA and MNBC correctly classified the packets in both instances as expected.

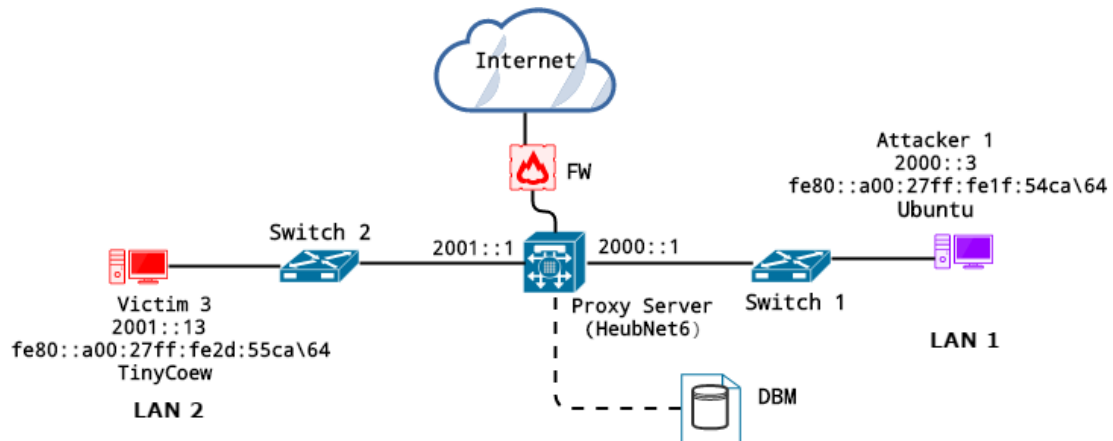


Figure 4.10: Parameter Problem anomaly detection topology

Table 4.12: Detection of Parameter problem covert data using NIHA and MNBC

Src (Alice)	LAN	Dst (Bob)	LAN	Value	Expected	NIHA	MNBC
Attacker 1	1	Victim 1	2	Type = 299	Covert	Covert	Covert
Attacker 2	2	Server	1	Type = 128	Normal	Normal	Covert

4.5.4 Computation of Class Label Using MNBC

This section shows an example of how MNBC computes packet covert classification to predict the class label (covert or normal) based on the Hop Limit field. Using the training data in Table 4.13, predict a class label using Naïve Bayes classification. The data tuples are described by attributes Traffic_class (TC), Flow_Label (FL), Hop_Limit (HP), Payload_Length (PL), ICMPv6_Type (T), ICMPv6_Code (C) and ICMPv6_Payload (PYL). The class label attribute covert channel has two values (Covert, Normal). The tuples were classified are (This is an example for a sample data with 8 attributes holding class but the captured data has nearly 80,000 packet instances),

Table 4.13: Training Data in Nominal Values to Naïve Bayes Format

Header	Selected Field	Input Value	Nominal Values	NB Format
Pseudo	Traffic Class	89 90	T_TC, F_TC	1233:1 1234:1
Pseudo	Flow Label	<i>src,port,dst,port,flow label</i> 2000::3 2001::19	T_FL F_FL	65:1 66:1
Pseudo	Hop Limit	456 457 458	HopL_INC, HopL_DEC, HopL_UNC	345:1 346:1 347:1
Pseudo	Payload Length	433 434 435	PLen_INC, PLen_DEC, PLen_UNC	100:1 101:1 102:1
Pseudo	Next Header	6 (frag) 22 (telnet)	Extra_CovHdr, No_Extra_CovHdr	85:1 86:1
Pseudo	Source Address	2001::7, 2001::13, 2001::15	Node_1, Node_2, Node_3	233:1 234:1 235:1
ICMPv6	Type & Code	1,0 255,255	Type 1_Code0 Type 255_Code255	45.1 65.378:1

$K = (TC = \text{false}, \text{true}, \text{Hop_Limit} = \text{unchanged}, \text{ICMPv6_Type} = 0, 1)$. If it needs to maximize $P(K|Ci)P(Ci)$ for $i = 1, 2, P(Ci)$, the prior probability of each

class can be composed based on the training tuples:

$K = (TC = \text{false}, \text{true}, \text{Hop_Limit} = \text{unchanged}, \text{ICMPv6_Type} = 0, 1)$. If a maximization is needed, then $P(K|C_i)P(C_i)$ for $i = 1, 2$, $P(C_i)$ the prior probability of each class can be composed based on the training tuples:

$P(\text{covert channel} = \text{Yes}) = 7/10 = 0.7$, $P(\text{covert channel} = \text{No}) = 3/10 = 0.3$

$$\text{Info}(S) = -\frac{7}{10}\log_2\left(\frac{7}{10}\right) - \frac{3}{10}\log_2\left(\frac{3}{10}\right) = \mathbf{0.881} \quad (4.1)$$

To calculate $P(K | C_i)$, for $i=1,2$, $P(C_i)$ the following conditional probability will be computed:

$P(TC = \text{false} | \text{covert channel} = \text{Yes}) = 2/10 = 0.2$
 $P(TC = \text{false} | \text{covert channel} = \text{No}) = 8/10 = 0.8$
 $P(HL = \text{unchanged} | \text{covert channel} = \text{yes}) = 1/10 = 0.1$
 $P(HL = \text{unchanged} | \text{Covert channel} = \text{No}) = 9/10 = 0.9$
 $P(\text{ICMPv6_Type} = 1 | \text{covert channel} = \text{Yes}) = 6/10 = 0.6$
 $P(\text{ICMPv6_Type} = 0 | \text{covert channel} = \text{No}) = 4/10 = 0.4$

Using the above probabilities, the following can be obtained: $P(K | \text{Covert channel} = \text{yes}) = P(TC = \text{false} | \text{covert channel} = \text{Yes}) * P(HL = \text{unchanged} | \text{covert channel} = \text{yes}) * P(\text{ICMPv6_Type} = 1 | \text{covert channel} = \text{Yes}) = 0.2 * 0.1 * 0.6 = 0.12$ and $P(K | \text{covert channels} = \text{No}) = 0.8 * 0.9 * 0.4 = 0.288$

To find the class C_i that maximizes $P(K | C_i)P(C_i)$, it can compute $P(K | \text{covert channel} = \text{yes}) * P(\text{covert channels} = \text{yes}) = 0.12 * 0.7 = 0.084$ $P(K | \text{covert channel} = \text{No}) * P(\text{covert channels} = \text{No}) = 0.3 * 0.288 = 0.0864$.

As mentioned previously that MNBC classifier predicts the covert channels = yes for tuple K . The classifier started to give a high ranking values of the attributes in the classification level as shown in Table 4.15 and a sample result of comparison between NIHA and MNBC is shown in Table 4.14.

To work out the decision tree information gain used on the given sample of data, the following calculations have been performed. Each value in the sample packet is a discrete value as the class label covert channel attribute has two distinct values (covert channel = Yes or No). Let us give them $m=2$, and let class C_1 correspond to Yes and class C_2 correspond to No, there are 7 tuples Yes and 3 tuples No. Let us create a root node N for the tuples in C_i , then to find the splitting criterion for these tuples, the information gain of each attribute must be computed using equation 3.3. As a sample of captured traffic, if 10 packets were taken with 8 attributes, each attributes have more than one subset values(tuples) as shown in Table 4.13, it can be calculated as follows:

$$Info(S) = -\frac{7}{10}\log_2\left(\frac{7}{10}\right) - \frac{3}{10}\log_2\left(\frac{3}{10}\right) = \mathbf{0.881} \quad (4.2)$$

Next, each attribute expected information requirement needs to be computed. Let us take attribute Hop_Limit category "unchanged" distribution, there are three No and one Yes; for "decreased " category, there are 5 Yes and 0 No; for "increased" category, there are 1 yes and 0 No. Using equation 3.4, it will be as follows:

$$\begin{aligned} InfoHopLimits(S) &= \frac{4}{10} * \left(-\frac{1}{4}\log_2\left(\frac{1}{4}\right) - \frac{3}{4}\log_2\left(\frac{3}{4}\right) \right) \\ &\quad + \frac{5}{10} * \left(-\frac{5}{5}\log_2\left(\frac{5}{5}\right) \right) \\ &\quad + \frac{1}{10} * \left(-\frac{1}{1}\log_2\left(\frac{1}{1}\right) \right) = \mathbf{0.324} \end{aligned} \quad (4.3)$$

The information gain of Hop Limit is **0.324 bit**. Next, let us take attribute Traffic Class category "True Class" distribution. There are four yes and three No. For "False Class" category, there are three Yes and 0 No, using equation 3.4, it

Table 4.14: Results of new attacks detection using NIHA & MNBC

Attack Name	MNBC Detection %	NIHA Detection %
Neptune (SYN Flood)	97.4	98.2
Xmas Tree	97.2	97.7
Multihop	97.9	97.9
Spy	97.8	98.5
Average Rate %	97.57	98.07

can be calculated as below:

$$\begin{aligned}
 InfoTraffic_{Class}(S) &= \frac{3}{10} * \left(-\frac{3}{3} \log_2\left(\frac{3}{3}\right) \right) \\
 &+ \frac{7}{10} * \left(-\frac{4}{7} \log_2\left(\frac{4}{7}\right) - \frac{3}{7} \log_2\left(\frac{3}{7}\right) \right) = \mathbf{0.846}
 \end{aligned} \tag{4.4}$$

The information gain of Traffic Class is **0.846** bit and the information gain of Hop Limit is **0.324 bit**. Next, let us take attribute Flow Label category "True Label" distribution. There are six yes and three No. For "False Label" category, there are one Yes and 0 No. Using equation 3.4, the calculation will be:

$$InfoFlow_{Label}(S) = \frac{9}{10} * \left(-\frac{6}{9} \log_2\left(\frac{6}{9}\right) - \frac{3}{9} \log_2\left(\frac{3}{9}\right) \right) = \mathbf{0.8264} \tag{4.5}$$

The information gain of Flow Label is **0.826 bit**. The other attributes will have the same calculations.

4.5.5 Exporter

The exporter produced audit data to verify the accuracy of the classifiers by using a new format of training and testing data ARFF files for cross validation against WEKA. The ARFF format file consisted of three headers: relation, attribute and data. Each word had the prefix @ as shown in Figure 4.11. The header section of

Table 4.15: Ranking of the attributes using C4.5 with InfoGain

Ranking	Att Sequence	Attribute Name
0.99	8	ICMPv6_PYL
0.55	3	HOP_Limit
0.55	4	Payload_L
0.33	5	Next_Header
0.33	2	Flow_Label
0.33	1	Traffic_Class
0.33	7	ICMPv6_Code
0.25	6	ICMPv6_Type

the ARFF file contained a list of the attributes (columns in the data), and their types. Thus, Table 3.5 has the @Relation as **IHATest**, the @attribute **Class** represents the detection classes {1 for Normal, 2 for Anomaly}, @attribute **Packet** represents the nominal representation of values in the IPv6 header fields, ICMPv6 header and extension headers. Lastly, the @data section in Table 3.6 was the normal transformed data from the network simulation language.

4.5.5.1 ARFF File

The ARFF format was produced after during Multinomial data transformation in order to be compatible for cross validation stage later and to be tested in WEKA. A sample of an ARFF data file is shown in Figure 4.11.

```

@relation 'IHATest'

@attribute class {1, 2}

@attribute traffic class      string    {t_tc, f_tc}
@attribute flow label        string    {t_fl, f_fl}
@attribute payload length    numeric    {plen_inc, plen_dec, plen_unc}
@attribute hop limit         numeric    {hopl_inc, hopl_dec, hopl_unc}
@attribute extension header  string    {extra_covhdr, no_extra_covhdr}
@attribute type              numeric    {typ1_code1}
@attribute code              numeric    {type0_code8}
@attribute icmpv6            string    {echo_req, echo_rep}

@data

2  3:1 2:1 6:1 25:2 5151:1 561:1 73:1 64654:1 74257:5
1  1:1 3:1 6:1 25:2 5152:1 560:1 72:1 17235:1 69235:2
1  1:1 3:1 6:1 34:2 4322:1 581:1 72:1 17235:1 69235:2
2  3:1 2:1 6:1 27:2 5151:1 545:1 86:1 64465:1 74344:5
2  3:1 2:1 6:1 26:2 5151:1 561:1 13:1 64345:1 74785:5
2  3:1 2:1 6:1 55:2 5151:1 558:1 25:1 64245:1 74567:5
1  1:1 3:1 6:1 23:2 5322:1 567:1 34:1 17235:1 69235:2
2  3:1 2:1 6:1 33:2 5151:1 556:1 64:1 64111:1 74212:5
2  3:1 2:1 6:1 11:2 5151:1 555:1 65:1 64222:1 74356:5
2  3:1 2:1 6:1 33:2 5151:1 544:1 55:1 64343:1 74345:5
2  3:1 2:1 6:1 25:2 5151:1 533:1 21:1 64121:1 74121:5

```

Figure 4.11: ARFF file format created for evaluation process

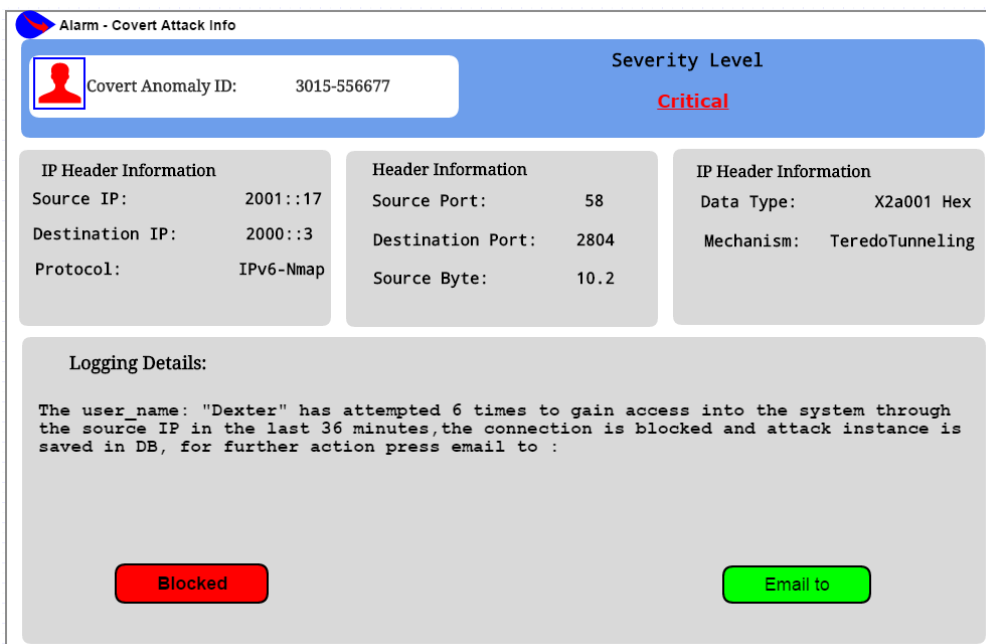
4.5.6 Alarm Flagger

When an anomaly is detected, the system creates a file log entry message to describe the detected anomaly in the IPv6 header. The alarm message consists of three sections:

- The first section shows the anomaly type via a packet ID representing a category either critical (severe level highlighted) or a warning. The system decides the severity type of the anomaly based on the new class label which is used to indicate whether a security policy violation has occurred, or an error has occurred.
- The second section contains various subsections of header details: destination and source, IP addresses, protocol number, source and

destination port number, amount transmitted source in byte (size), data type sent and mechanism used (tunnelling, teredo etc.).

- The third section contains logging details about the attempted service name (user_ID), frequencies occurrences tried to gain access, last attempts and suggesting to email/export the log entry/file if needed for further analysis. Additionally, the group of anomaly will be shown with regards to which network domain was attacked via an intelligent shell script. Figure 4.12 shows a sample of a generated alarm when an anomaly covert is detected.



Alarm - Covert Attack Info

Covert Anomaly ID: 3015-556677

Severity Level: **Critical**

IP Header Information	Header Information	IP Header Information
Source IP: 2001::17	Source Port: 58	Data Type: X2a001 Hex
Destination IP: 2000::3	Destination Port: 2804	Mechanism: TeredoTunneling
Protocol: IPv6-Nmap	Source Byte: 10.2	

Logging Details:

The user_name: "Dexter" has attempted 6 times to gain access into the system through the source IP in the last 36 minutes, the connection is blocked and attack instance is saved in DB, for further action press email to :

Blocked **Email to**

Figure 4.12: Alarm notification of an anomaly detection from HeuBNet6

4.6 Summary

In this chapter, 37 experimental tests were performed on the investigated header field attributes. All instances of covert channels' attacks have been performed successfully. The performance was obvious throughout all experiments and testing phases. The test cases were planned and scheduled along with an excerpt of the designed network topology for this project. Some challenges were faced during the simulation attacks operations. Usually, complying with the main scenario always creates a challenge once network design is initiated. In spite of the setbacks, results from the experiments indicated that the model operated and performed correctly. This progress was achieved through obtaining the preliminary and primary data needed for experimental covert detection and classification. The live anomaly detection process of covert channels in IPv6 was functioning smoothly dealing with large traffic throughput. A multi-threaded queue mechanism was deployed to deal with a high speed which was the highest network that could be afforded to split the gathered traffic into two queues over many designated IP addresses within the TCP/IP protocol. Thus, the multiple security and detection process using NIHA and MNBC achieved good results offering high detection rate with an overall of **98.%** and low false positive rate (FPR). Further analysis of experimental results are shown in chapter 5.

Chapter 5

Evaluation Methodology and Analysis

In Chapter 4, the empirical implementation and testing of the model was performed. This implementation led to create potential novel primary data through using two essential modules: Network Intelligent Heuristic Algorithm (NIHA) and Multinomial Naïve Bayes Classifier (MNBC). On the other hand, this data consists of two types of datasets: training and testing datasets which will be used in the holdout cross validation method later. The achieved results are impressive with regards to the significant accuracy which obtained detection rate 96.66% of the overall process with good performance. New suggested models and systems always need to be validated and evaluated against other similar systems and other synthetic data. According to the methodology which was conducted in this research, one of the outcome components of the suggested model is the primary data, which is unique and unlikely to be validated against KDD-NSL benchmark data. This is due to the incompatibility in the format type and their features held in comparison to DARPA data type. In this chapter, vital elements in validation and evaluation of the suggested model are discussed.

Basically, decision making in network security depends on creating or gaining data from its original sources and this is considered to be the essence of networking operations. This data should be reliable because it is completely collected during data acquisition period. Furthermore, the data should be accurate and consistent

based on the reliable testing measures. Finally, the data should be flexible and compatible for future testing and examination by researchers.

This chapter is organised as follows: Section 5.1 discusses the cross validation; Section 5.2 explains evaluation metrics; Section 5.3 discusses the confusion metrics; Section 5.4 discusses the precision analysis; Section 5.5 discusses the covert detection evaluation; Section 5.6 presents a discussion about critical analysis; Section 5.7 discusses the weaknesses of Naïve Bayes algorithm; Section 5.8 presents a summary of the evaluation results; Section 5.9 summarises the chapter.

5.1 Cross Validation

In order to obtain a reliable accuracy estimate, standard measures should be specified for the classifier through cross validation process [171]. Some of the data is removed before training begins. Then when training is done, the removed data can be used to test the performance of the learned model on "new" data. This is the basic idea for a whole class of model evaluation methods called cross validation.

The holdout method [172] as shown in Figure 5.1 is the simplest type of cross validation. In this method, the given data are randomly partitioned into two independent sets, a training set and a test set. Essentially, two-thirds of the data are allocated to the training set, and the remaining one-third is allocated to the test set. The training set is used to derive the model, then the model accuracy will be estimated with the test set. Apparently, the estimate will be dealt only with one portion of the initial data used to derive the model. The function approximator fits a function using the training set only. Then the function approximator is asked to predict the output values for the data in the testing set. The output errors are accumulated as occurred before to give the mean absolute test dataset error, which obviously is used to evaluate the model.

The advantage of this method is that it is usually preferable to the residual method and takes no longer to compute with a high variance. There are other cross validation methods such as k-fold and leave-one-out (LOOCV) cross validation [8, 172]. k-fold is useful when no dataset is available which will be used with

the new primary data. Leave-one-out is very costly due to the large amount of outcome data in this project.

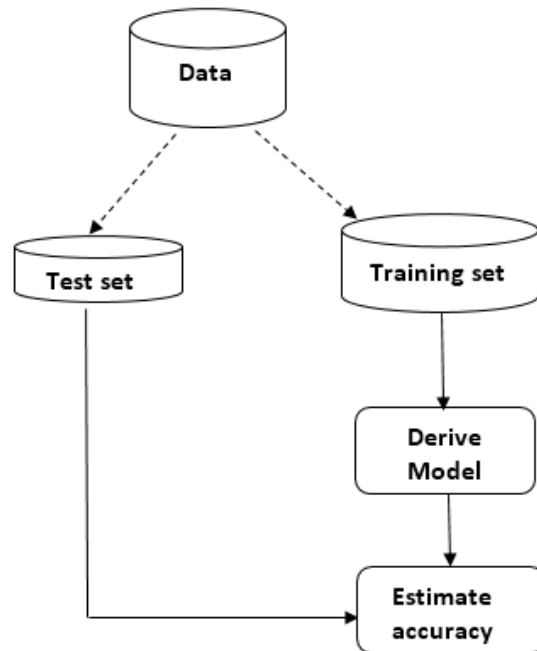


Figure 5.1: Estimation accuracy with holdout method [8]

To perform an evaluation of the system, WEKA 3.7 database system built in Java programming language has been used on a personal computer with 3.1 GHz Intel core i5 CPU 3450 and 8 GB RAM. To provide an evaluation of the proposed framework, a deep analysis of the detection modules of NIHA and MNBC has been conducted, and sufficient results of evaluation have been obtained. The dynamic analysis of the framework via validation and verification of the created primary data by the system has been performed.

The results of all testing experiments show a high accuracy of initial hypothesis performed by NIHA and MNBC. Table 5.1 and Figure 5.2 show a distinguished correctness with a low false positive of the new classifier. The detection rate (DR) was 98%.

NIHA was run to process the detection of selected characteristics of covert

Table 5.1: Overall HeuBNet6 Accuracy Detection

Classifiers	Accuracy	TPR	FPR	Precision	Time
Naïve Bayes	86%	76%	0.23	0.76	1.50
NB+InfoGain	65%	83%	0.38	0.82	1.40
HeuBNet6	98%	98%	0.12	0.98	1.20
NB+SubSetEval	55%	81%	0.32	0.74	1.45
SVM	96%	94%	0.15	0.94	1.25
C4.5	91%	93%	0.15	0.89	1.30
GA	97%	95%	0.14	0.93	1.23

channels in IPv6. Additionally, processing the classification module resulted in better performance with regards high detection rate. All experiments have been performed on the original dataset, with the included eight attributes. HeuBNet6 has been compared to the following built-in classifiers in WEKA: Naïve Bayes classifier, Subset Evaluation Technique, Bayes Net, NBC, InfoGain, SVM (Gaussian process), C4.5 and GA. In Appendix A.4, Table 2 and Table 3 show the HeuBNet6 primary data format that have been used in cross validation process. Figure 5.2 and Table 5.1 show an overall accuracy gained from testing NIHA against other current classifiers.

Table 5.2: Sample training dataset used in Evaluation Process

Class	TF	FL	PL	NH	SA	DA	TY	CD	PY
anomaly	1	99	47	60	2001::7	2000::3	128	8	payload
anomaly	1	99	47	43	2001::7	2000::3	128	8	payload
anomaly	1	99	47	44	2001::7	2000::3	128	8	payload
anomaly	1	99	47	60	2001::7	2000::3	128	8	payload
anomaly	1	99	47	43	2001::7	2000::3	128	8	payload

5.2 Evaluation Metrics

The performance of MNB classifier was evaluated using the detection rate, false alarm rate and overall accuracy. Generally, targeted metrics and measures to

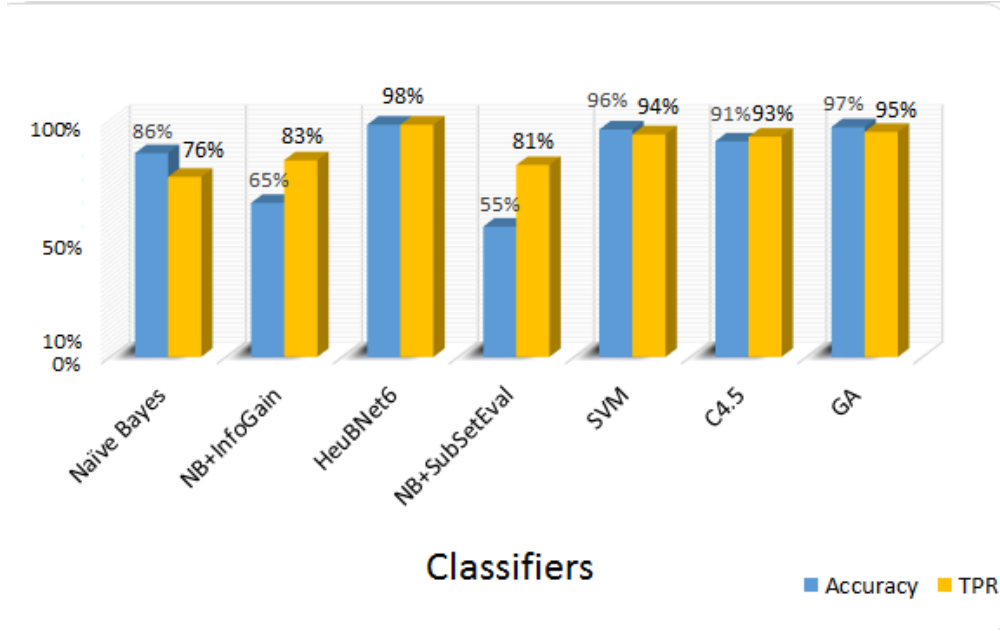


Figure 5.2: Overall HeuBNet6 accuracy in comparison to other classifiers

evaluate the correctness of the classification module are used as "building blocks" for covert channels detection. These are defined as follows [6]:

- True Positive (TP): Number of correctly identified positive tuples of covert channels. The overall TP are 78,800.
- True Negative (TN): Number of correctly labelled normal tuples by the classifier. The overall TN are 7,630.
- False Positive (FP): Number of incorrectly labelled normal tuples by the classifier. The overall FP are only 1110.
- False Negative (FN): Number of wrongly labelled covert channels tuples by the classifier. The overall FN are 1130.

The detection rate of any specified attack can be measured by the following metrics notations and their calculations:

1. Accuracy formula is

$$\frac{TP + TN}{TP + TN + FP + FN} * 100 \quad (5.1)$$

The result of calculation is:

$$\frac{78800 + 7630}{78800 + 7630 + 1130 + 1110} * 100 = 97\% \quad (5.2)$$

2. Detection rate is

$$\frac{TP}{TP + FN} * 100 \quad (5.3)$$

The detection rate will be:

$$\frac{78800}{78800 + 1110} * 100 = 98.61\% \quad (5.4)$$

3. False alarm is

$$\frac{FP}{FP + TN} \quad (5.5)$$

The false alarm rate is:

$$\frac{1110}{1110 + 7630} = 0.12\% \quad (5.6)$$

5.3 Confusion Matrix

The confusion matrix as shown in Table 5.4 is a useful tool to evaluate the performance of the suggested classifier in recognizing tuples of different classes. The numbers of correct and incorrect detections are presented into the columns of the confusion matrix generated by the classification model. The metrics is NxN, where N is the number of target values (classes). MNBC data collected through WEKA are presented in Table 5.2. Figure 5.3 shows the results of the metric measures of the HeuBNet6. The data sample in Table 5.2 was used in the cross validation process. The sample consisted of ten columns, each column holding the characteristics of covert data type and its subset values. The columns headers were: class (1 for anomaly, and 2 for normal), traffic class (TF), flow label (FL), Payload length (PL), next header (NH), source address (SA), destination address (DA), type (TY), code (CD), and finally payload (PY). NIHA has been evaluated through creating two criteria as will be explained in Section 5.5. In this case the classifier is evaluated in detecting covert channels based on features of IPv6 and ICMPv6 headers. The ability of this classifier is shown in evaluation using different performance measures criteria such as

5. Evaluation and Results Analysis

accuracy, sensitivity, specificity, and Receiver Operator Characteristics (ROC) curve. Obviously, the MNBC classifier has performed exclusively correct in comparison to other classifiers as shown in Table 5.1 and Figure 5.3. The overall detection rate of HeuBNet6 was 98% which is the optimal accuracy and performance can be obtained.

Table 5.3: MNBC Data Size Used in Confusion Matrix for covert anomaly detection

Classifier	TP	TN	FP	FN
MNBC	78800	7630	1130	1110

Table 5.4: Confusion Matrix Showing the Actual Predicted Class

Classes	Predicted class		Total
	Predicted normal	Predicted covert	
Actual no	TN	FP	N
Actual yes	FN	TP	P
Total	N'	P'	N+P

In order to work out the complete computation of the values which have been obtained from the classifier, the given values in confusion matrix Table 5.5 is used. The following calculations have been made:

1. Sensitivity or True Positive Rate (TPR) is to find out how often the actual anomaly class is predicted. This is done as follows:

$$\frac{TP}{actual_yes} * 100 = \frac{78800}{79930} * 100 = 98\% \quad (5.7)$$

2. Misclassification (Error Rate) is to find out how often the classifier predicts it wrong. This is done as follows:

$$\frac{FP + FN}{total} * 100 = \frac{1110 + 1130}{88670} * 100 = 2.5\% \quad (5.8)$$

3. Specificity is correct prediction of actual "no" by the classifier as shown in:

$$\frac{TN}{actualno} * 100 = \frac{7630}{8740} * 100 = 87\% \quad (5.9)$$

5. Evaluation and Results Analysis

4. Precision is the frequency of the "yes" prediction by the classifier. It is done as follows:

$$\frac{TP}{predicted\ yes} * 100 = \frac{78800}{79910} * 100 = 98\% \quad (5.10)$$

5. Prevalence is working out how often the "yes" class condition actually occurs in the test dataset. It is done through:

$$\frac{actual\ yes}{total} * 100 = \frac{79930}{88670} * 100 = 90\% \quad (5.11)$$

Table 5.5: Values of Confusion Matrix with the Actual Predicted Class

Classes	Predicted class		Total
	Predicted normal	Predicted covert	
Actual normal	7630	1110	8740
Actual covert	1130	78800	79930
Total	8760	79910	88670

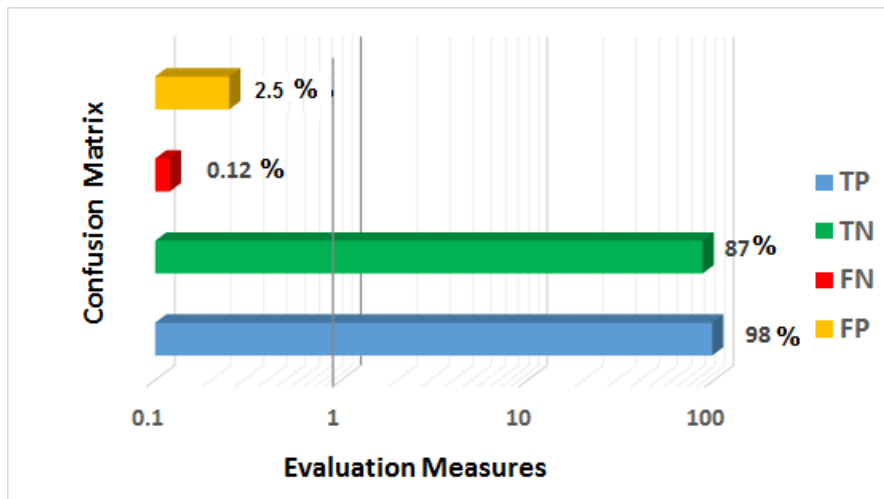


Figure 5.3: HeuBNet6 Confusion Matrix

5.4 Precision Analysis

The sensitivity and specificity are the conditional probabilities. The prevalence is the prior, and the positive/negative predicted values are the posterior

5. Evaluation and Results Analysis

probabilities. The suggested decision tree C4.5 created a positive power along with Multinomial Naïve Bayes algorithm on the detection rate with a precision accuracy of $0.98 \times 100 = 98\%$ as shown in Table 5.1 and Figure 5.2. Apparently, an obvious improvement with 98.61% is observed in comparison to the first experimental results testing similar to implemented techniques. In order to create a more reliable experiment result, a 10-fold cross validation was performed on HeuBNet6 through dividing the whole primary data into 10 partitions of nearly equal sizes. The results are shown in Table 5.6 and the ROC curves are shown in Figure 5.4. The positive predicted values of both NIHA and MNBC modules as shown in Figure 5.2 are elaborating an impressive enhancement in HeuBNet6 performance.

Table 5.6: Sensitivity and Specificity of HeuBNet6 performing 10-Fold Cross Validation

TP	0.98	0.97	0.96	0.95	0.94	0.93	0.92	0.91	0.89	0.86
TN	0.87	0.86	0.85	0.84	0.84	0.83	0.82	0.80	0.79	0.79

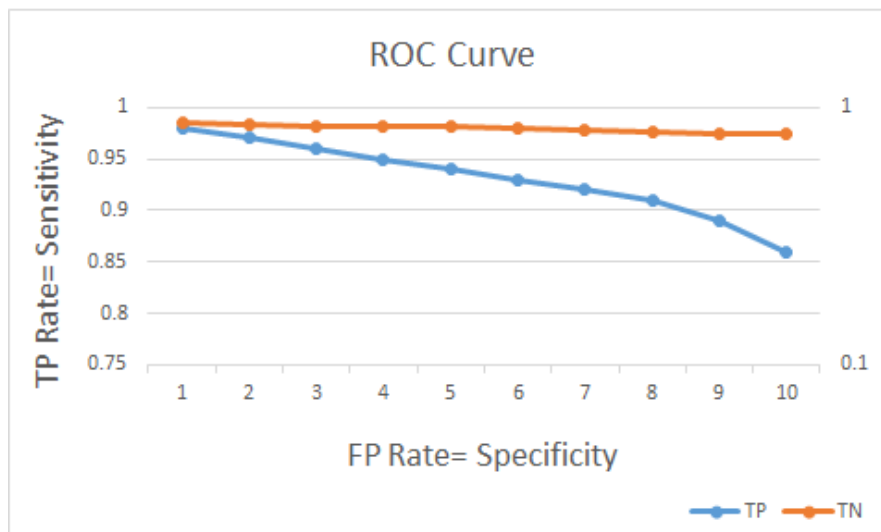


Figure 5.4: Performance of precision Level of MNBC

The Precision accuracy outcomes show the reliability of the experiment results. To obtain a statistical evaluation by reading the final results of all

5. Evaluation and Results Analysis

Table 5.7: Precision of HeuBNet6 Model Accuracy in Comparison to Current Classifiers

Classifiers	Accuracy	TPR	FPR	Precision	Time (ms)
Naïve Bayes	86 %	76%	0.23	0.76	1.50
NB+InforGain	65%	83%	0.38	0.82	1.40
NB+SubsetEval	55%	81%	0.32	0.74	1.45
SVM	96%	94%	0.15	0.94	1.25
C4.5	91%	93%	0.15	0.89	1.30
GA	97%	95%	0.14	0.93	1.23
HeuBNet6 (MNBC)	98%	98%	0.12	0.98	1.20

classifiers and their performance against HeuBNet6, the suggested classifier precision can be observed as 0.98, which is higher than subset evaluation algorithm 0.74 and the precision of InfoGain with NB was 0.82. Furthermore, SVM using Gaussian function which is the compatible technique to the primary data obtained 0.94 with accuracy of 96%. Genetic Algorithm exposed with the next highest accuracy of 97% and a precision of 0.93.

In analysing Table 5.7, a significant performance of the MNBC can be observed in both elements: time spent and precision rate. MNBC has given the highest value of precision rate with 0.98% accompanied with lowest time elapsed 1.20 ms. If compare these two gained values with Naïve Bayes which obtained 0.76 elapsing 1.50 ms, NB supported with InfoGain achieved 0.82 precision accuracy and the time elapsed was 1.40 ms. NB+SubSetEval technique gathered 0.74 and time elapsed was 1.45 ms. If the SVM precision compared with HeuBNet6 (MNBC) performance, a difference of 0.4% is found as SVM obtained a lower accuracy rate of 2% elapsing difference of 0.5 ms. Finally, C4.5 individually was tested to evaluate the novel data. The accuracy result was 91% with a precision rate of 0.89 and a difference of elapsed time to create the model is 0.10 ms. So obviously, in terms of the initial detection performance and final results, HeuBNet6 is the fastest that obtained the highest accurate covert channel detector among the other advanced machine learning methods.

5.5 Evaluation of Covert Detection

The system HeuBNet6 produces four components in a sequential manner: alarm message, statistical report, analytical report and evaluation report. These reports contain different data types and figures. Evaluation report contains the covert channel characteristics outcome from the targeted values. This type of file would theoretically be run into a similar system which can handle with the same created data type (attack instances).

Figure 5.5 is a sample of NIHA and MNBC output classification data with class type and accuracy rate.

```
NIHA NB-Format: 1 9:1 26:1 6:1 8:1 11:1 15:1 2045:1
MNBC Classifying packet: 19:1 26:1 6:1 8:1 11:1 15:1 2045:1
MNBC pred_class: 1
MNBC Accuracy: 0.98
```

Figure 5.5: A sample of NB classified output data format

It shows the accuracy calculated from the detection classes of every immediate 20 processed packets. The accuracy is stored and averaged over a set detection period. The overall detection rate of HeuBNet6 was roughly 97.61% as compared to the 98.41% accuracy obtained by WEKA using the NB-Formatted test data converted from the packet values as shown in Figure 5.5. HeuBNet6's reduction in detection rate could be attributed to lost packets during test runs either by the packet crafting tool or from the Libnetfilter queue.

The cross validation process in WEKA was useful and verified the gained results and hypothesised scenarios. This is particularly important because no other IPv6 covert detection tools are available to test the new generated datasets since similar tools are proprietary and unpublicized.

5.5.1 Covert Detection Criteria

This section shows the evaluation of some of the covert detections based on the selected IPv6 pseudo header and ICMPv6 headers experimented in this research.

Each evaluation calculates the experimented threat level of each attribute which was selected.

The attributes have been divided into two criteria: The first criteria is Cr-1 contains all attributes which hold more than one subset value, as an opportunity to test the vulnerability of packets based on a particular subset value. The second criteria is Cr-2 contains all attributes with one value for each, hence is not possible to be clearly evaluated as a key attribute (Key).

1. Echo Request Anomaly Detection Evaluation

This evaluation aims at identifying the level of threat posed by echo request messages. Echo request messages could be exploited to act as a Ping command to discover computers on the network by an attacker. The covert attack packets could have any of the valid IPv6 next header protocol numbers to allow further covert attacks in the extension header. This evaluation shows that HeuBNet6 correctly identified 2,304 anomalous echo request packets between two computers out of 2,305 overall samples.

- Selected Attributes
 - Type = 128 (Cr-1) (Key)
 - NH = ALL (Cr-1) (Key)
 - SA=2001::7 (Cr-2)
 - DA=2000::3 (Cr-2)
- Results
 - Total Covert = 2,304
 - Total Normal = 1
 - Total Samples = 2,305

Figure 5.6 shows a medium number of covert attacks associated with Echo Request Messages (Type = 128). The type-code attribute has a covert-normal ratio of 128:1.

2. Echo Request with ICMPv6 Detection Evaluation

This evaluation analyses the echo request covert attacks that specifically

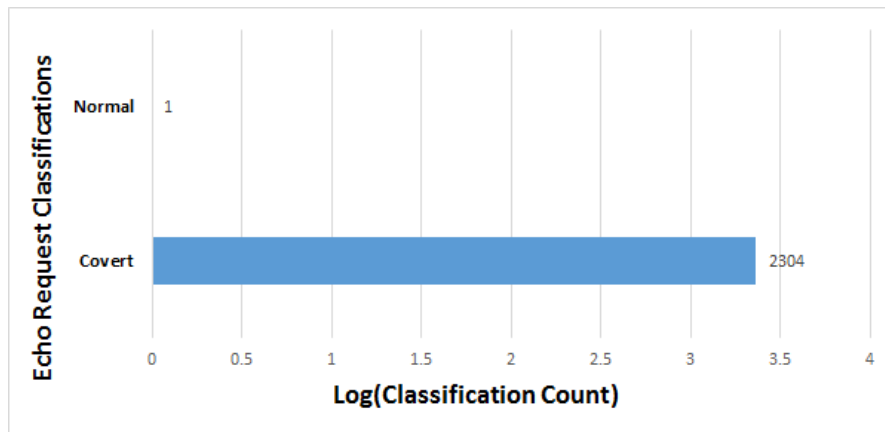


Figure 5.6: Echo Request evaluation graph

carry covert data in the ICMPv6 Extension header. 256 out of 258 tests were classified as covert attacks.

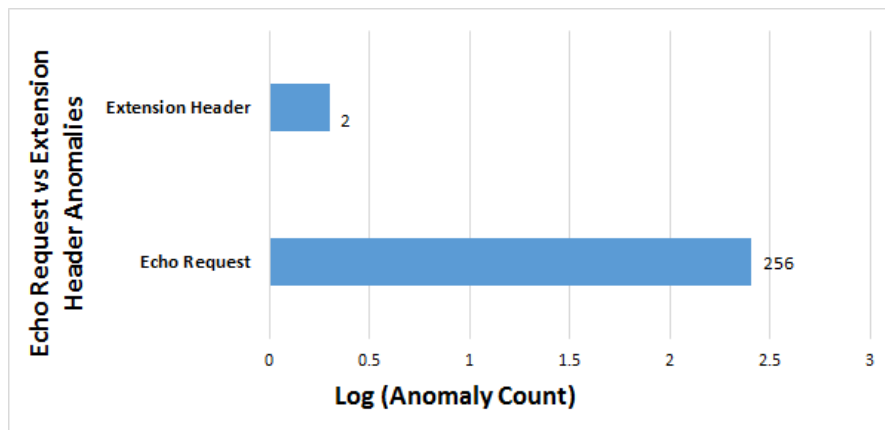


Figure 5.7: Echo request evaluation logarithmic graph

- Selected Attributes
 - Type = **128** (Cr-1) (Key)
 - NH = **58** (Cr-1) (Key)
 - SA=2001::7 (Cr-2)
 - DA=2000::3 (Cr-2)

- FL = 99 (Cr-2)
- Results
 - Total Samples = 258
 - Echo Request = 256
 - Extension Header = 2

Figure 5.7 shows a high number of covert attacks associated with Next header (58) and Echo Request Messages (Type = 128). The covert-normal ratio for echo request with ICMPv6 extension header attacks is 128:1.

3. ICMPv6 Extension Header Covert Detection Evaluation

This evaluation analyses ICMPv6 extension header covert attack detection since the ICMPv6 header is of utmost importance in the IPv6 protocol. From the 80,000 training and testing samples as shown in Appendix A.4, 7940 packets from the source address (2001::7) to the destination address (2000::3) were classified as normal while only 766 as covert.

- Selected Attributes
 - Type = 128 (Cr-1) (Key)
 - NH = 58 (Cr-1) (Key)
 - SA=2001::7 (Cr-2)
 - DA=2000::3 (Cr-2)
 - FL = 99 (Cr-2)
- Results
 - Total Samples = 8,706
 - Total Covert = 766
 - Total Normal = 7,940

Figure 5.8 shows the number of covert attacks associated with normal ICMPv6 Next Header extension Message (NH = 58).

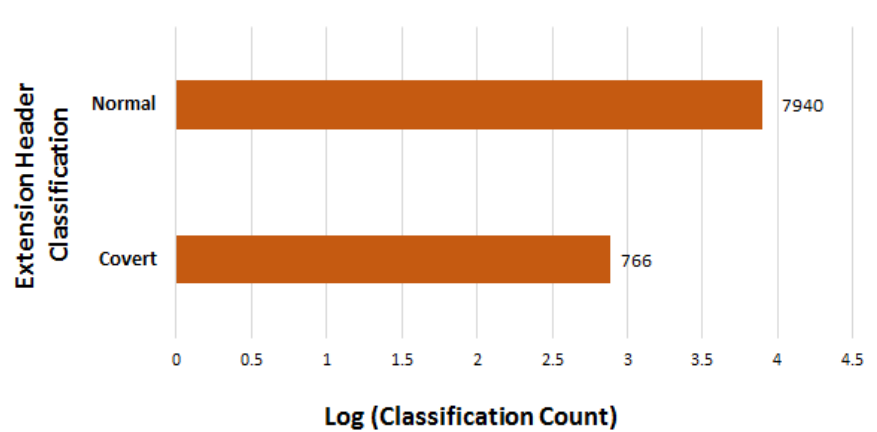


Figure 5.8: ICMPv6 Header Covert Detection graph

4. ICMPv6 Extension Header vs Other IPv6 Extension Headers

This section compares ICMPv6 extension header covert attacks against the rest of the IPv6 extension headers {6, 43, 44, 50, 51, 59, 60,135}. It was found that ICMPv6 extension header had 766 normals while the rest of the extension headers had 8,662 anomalies. The anomalies for the other extensions were considerably high due to some incomplete experimental implementation. It should be also noted that two further classifications namely DROP and ALLOW are suggested in the pseudo-algorithms to handle cases. This is while packets are supposed to be dropped or allowed regardless of their classification. These classifications together with the completion of the proposed algorithm would change the preliminary results.

- Selected Attributes
 - NH = **58** (Cr-1) (Key)
 - SA=2001::7 (Cr-2)
 - DA=2000::3 (Cr-2)
 - FL = 99 (Cr-2)

- Results
 - Total Samples =9428

- Total Covert = 8662
- Total Normal = 766

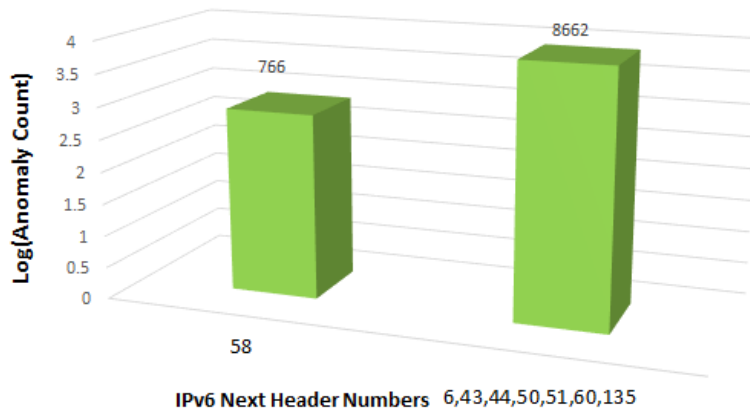


Figure 5.9: ICMPv6 Next Header evaluation logarithmic graph

Figure 5.9 shows a comparison of experimental covert attacks in the ICMPv6 extension header and the rest of the IPv6 extension headers {6, 43, 44, 50, 51, 59, 60,135} for packets sent between the two nodes listed above.

5. Echo Request Anomaly vs Next Header Anomaly Evaluation

Figure 5.10 shows the comparison between echo request anomalies against next header anomalies. It is observed that there are 2,304 echo request anomalies while there are only 766 normals for next header.

- Selected Attributes
 - NH = 58 (Cr-1) (Key)
 - SA=2001::7 (Cr-2)
 - DA=2000::3 (Cr-2)
 - FL = 99 (Cr-2)
- Results
 - Total Samples = 3068
 - Total Covert = 2304

– Total Normal = 766

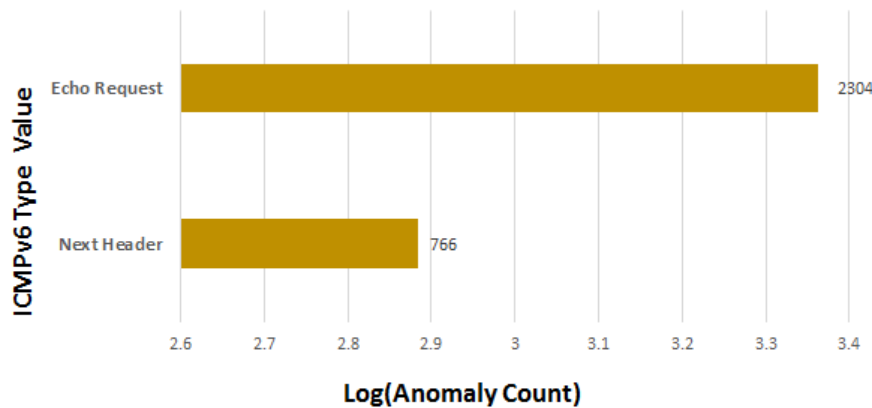


Figure 5.10: Echo Request Anomaly versus Next Header Anomaly Evaluation logarithmic graph

6. Echo Request and Multicast Listener Discovery (MLD) Anomalies Detection Evaluation

This section shows a comparison between anomalies based on multicast listening discovery and echo request messages. This is an important metric for detecting whether a source node is probing the network to discover other nodes with a probable intention to carry out attack on it or a node is legitimately broadcasting many solicitation messages. In case of an event that an attacker is pinging many nodes on the network, HeuBNet6 would clearly detect the discrepancy in the rate of solicitation messages and echo request messages.

- Selected Attributes
 - Type = 128 (Cr-1) (Key)
 - Type = 130 (Cr-1) (Key)
 - SA=2001::7 (Cr-2)
 - DA=2000::3 (Cr-2)
 - FL = 99 (Cr-2)
- Results

- Total Samples = 4353
- Total Covert = 2304
- Total Normal = 2049

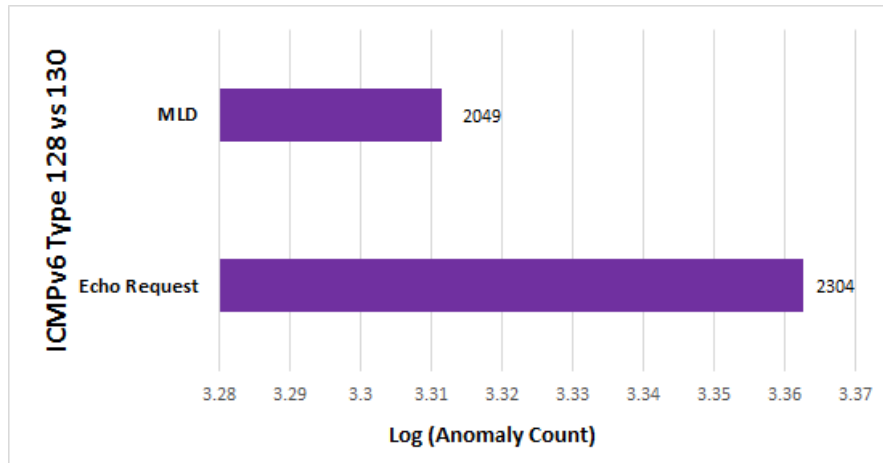


Figure 5.11: ICMPv6 Next Header evaluation logarithmic graph

Figure 5.11 shows that there were proportionally equal anomalies in echo request messages and multicast listener messages in the HeuBNet6 covert detection experimentation.

5.6 Critical Analysis

This Section provides critical analysis of the model performance with regards to the detection functionalities and other issues. Generally, countermeasure processes can not remove hidden information in steganography, which should be detected, analysed, eliminated or documented instead. The suggested method can be applied in passive warden threat model and active warden countermeasure model, despite the fact that active warden is unable to remove all steganography in the network lower level and the application model upper level [5, 15, 43, 46]. The application layer detection process will damage the information carrier and will not be successful [37] to eliminate all covert channels anomaly implications. The multi-layer security suggested in this multi-threaded

approach has not performed accurately to gain the ultimate level of optimization due to the unlikely controlled experiment environment. This environment lacked the possibility to function on live corporate business network. Conversely, HeuBNet6 behaved and performed perfectly as shown in Figure 5.2 corresponding to the correct values in Table 5.7 in relation to detection accuracy, in addition to the comparison among other different algorithms.

5.6.1 Detection Method

Commonly, as mentioned in Chapter 2, there are two main detection methods referred to as misuse (knowledge based) and anomaly (behavioural based) detection [114]. The first attempt, known as signature based IDS, encodes knowledge of known intrusions (misuses) typically to rules, and uses them to screen events. The second attempt tries to learn the features of each event and its patterns which constitute normal behaviour, then through observing these patterns that are distinguished from other established norms, detect an intrusion when it occurs [173].

Some of these IDSs offer both capabilities, typically via a hybridisation (heterogeneous) of techniques [114, 174]. Xiang *et al* [175] suggested a hybrid system which may be modelled according to both normal and intrusive data, which later has become a common approach in recent research adopting machine learning techniques [114].

Additionally, this general perception about misuse detection is no longer entirely accurate. In recent years, flexibility has become one of the vital features that researchers have attempted to create into incorporated techniques, to allow misuse detection systems to be capable of detecting more variations of attacks. This case offered the possibility to implement machine learning techniques along with rule based systems to detect variations of attacks, which were deemed in the past to be unable to detect even slight variations of attacks due to rigid rules [176]. In addition to the capability of Rule Based Systems (RBS) of detecting variations of attacks, they may even be deployed for anomaly detection. Researchers incorporated fuzzy logic and genetic algorithm to define the rules and successfully managed to use them in IDS [21].

An intelligent algorithm based detection system is applied in order to create a new adaptive rule based detection framework to investigate the association level among the header fields usability by the attacker when misusing the oversight design vulnerability in this protocol IPv6 [132]. Hold out method was used to cross validate the model which gave a higher variance than k-fold technique. The main reason is that the k-fold cross-validation estimator has a lower variance than a single hold-out set estimator, which can be very important if the amount of data available is limited. Obviously, there will be a lot of variations in the performance estimate for different samples of data, or for different partitions of the data to form training and test sets. k-fold validation reduces this variance by averaging over k different partitions, so the performance estimate is less sensitive to the partitioning of the data. Moreover, further repeated k-fold cross-validation can be done, where the cross-validation is performed using different partitioning of the data to form k sub-sets, and then taking the average over that as well in addition to the possibility of costly validation for such high volume of data.

5.6.2 HeuBNet6 as a Rule Based System

Expert system is one of the most common form of rule based systems which can be applied to detect intrusions. It uses event correlation (association) to detect misuse attack type, and this is a significant sign of RBS. The suggested system developed an intelligent rule based concept such as *IF-THEN* rules to recognize covert channel and new unknown attacks. These extracting rules can be used directly in the training dataset without having to generate a decision tree. This rule based algorithm can be used in parallel to a non rule based algorithm as an additional security thread using Multinomial Bayes classifier. The framework is flexible and can be installed on end point systems such as hosts to be a part of host based anomaly detection, then it will focus on user behaviour or process/program behaviour. In this case, the system relies on the pattern which relates to the user behaviour built to several profiles for users, providing them with access privileges. However, a major challenge of host based anomaly detection systems is that the system needs to keep up to date with environmental changes which eventually will increase false alarm due to behavioural drift [177]. This technique can cause an

issue with the awareness of the users in the anomaly training, then they gradually change their behaviour which will leave no impact on detection mode [178].

5.6.3 HeuBNet6 as a Hybrid System

As mentioned above, hybridisation includes intelligent rules and is possible to be deployed in different types of IDSs. There are some features of this technique which can be summarised as:

1. **Algorithmic:** in this stage, techniques or algorithms are hybridised to perform a single task together compounding a system such as using genetic algorithms to evolve rules [21, 179].
2. **Hierarchical:** in this stage, there is a hierarchy observed in the architecture of the IDS, which includes various techniques performing multi tasks in different ways at each level. An example of this is deploying RBS in a high level module to obtain correlated alerts from a lower module detector.
3. **Cooperative:** in this stage, different techniques are suggested to perform multiple and independent tasks, which later will be combined somehow to form a holistic system such as designating one technique to perform misuse detection and another to perform anomaly detection [24, 46].

The suggested security system operates like many other systems with combinations of all three mentioned categories. However, there are several examples of the first two categories discussed previously [141]. Basically, the IDS deploys an RBS for misuse detection and uses fuzzy association rule mining for anomaly detection [21]. Additionally, the anomaly detection enables the direction to build hybridization of association rule mining with GA to optimize fuzzy element functions, meanwhile a GA is basically used to perform feature selection.

5.6.4 Statistical Report

In addition to the evaluation report produced by the system and sent to cross validation process, the system comprises useful statistics details of the generated

data for business and professional needs. The statistical reports produced by the system involves the following items:

1. Attribute score predictive analysis.
2. Attribute probability analysis.
3. Detection Rates (DR) for True Positive Rate (TPR), False Positive Rate (FPR) and True Negative Rate (TNR).
4. Classification accuracy for MNBC and NIHA.
5. Liable Covert Attacks in features comparison and analysis.
6. Statistical Figures.

5.6.5 Analytical Report

Analytical reports can provide the means of assessing various aspects of the proposed system for proof of correctness, performance assessment and recommendations for future work. The analysis report contains the following items:

1. Behaviour of the NIHA in processing large size of data.
2. Performance of MNBC in compare to NB.
3. Time elapsed in detection and accuracy rate for NIHA and MNBC.
4. True Positive Rate (TPR), False Positive Rate (FPR), True Negative Rate (TNR), Detection Rate (DR) analysis.
5. Confusion Matrix Figures.

5.7 Weaknesses of Naïve Bayes Algorithm

In Section 3.7 the suggested hypothesis Bayesian classifier was used as a supervised learning technique. This technique allowed us to determine uncertainty of the model by determining probabilities of interdependent events, furthermore, assigning the attributes that exist in the IPv6 header, ICMPv6 header and the selected Upper Layer Protocol UDP from TCP header. The classifier is used due to the flexibility of its diagnostic technique and prediction of covert channels where number of input parameters were engaged.

The probability of an attribute in belonging to a classified category has been maximized such as covert or normal depending on a novel training dataset. In other words, this system presented a new approach to predicting multiple classes for tuples based on conditional probabilities of the data set. However, Naïve Bayes essentially has some weaknesses which may produce unexpected results or outcomes from the experimented testing and examinations of the selected and extracted features [180]. The significant challenges that the suggested technique attempted to resolve, are as follows:

1. Difficulty in getting most of a priori conditional and joint probabilities is expected in the hypotheses. This issue was solved using normalization techniques through dissecting the attributes subset types and values included in the extracted header fields from the messages. These values were transformed to numerical and human readable language, used in the data pre-processing phase then passed to NIHA as well as into the suggested MNBC algorithm to create sufficient data types and results.
2. Managing the huge content in the databases was slightly hard to control. This was due to the large amount of captured packets during the training and active warden phases collecting raw data, pre-processing data, then storing them into the database for processing and producing the suggested reports by the exporter process.
3. This type of mechanism needs a lot of calculations. A multi-threaded approach is proposed through using Object Oriented Programming

language C++. This programming language offered a decent speed in calculations.

4. The outcome should be separated in order to distinguish the classes and expected values which must be validated using other benchmark results. This issue was overcome through an evaluation stage using WEKA tool to compare the prediction of classification based on the new training datasets created by NIHA with reference to all known IPv6 RFCs. The results were exactly similar to the outputs produced from HeuBNet6 security model.

5.8 Evaluation Results

In this section, a conclusion of the evaluation process is presented. In cross validation, a holdout and k-fold were used in order to evaluate the suggested system performance. In the initial stages of the experiment, the HeuBNet6 detection rate was 97% operating on the new data created by NIHA using cross validation holdout technique. This technique divides the data into two sets: training dataset and testing dataset. The time elapsed in this process was 1.20 ms and the precision was 0.98 which is quite impressive if compared with other similar techniques. The prevalence of NIHA was 90%, showing the highest correctness occurrence of covert class condition among the other classifiers. The precision rate of NIHA and MNBC was 0.98, and the highest accuracy rate was obtained through 10-fold cross validation. Meanwhile, the lowest sensitivity rate of NIHA is 0.86. This is the best optimal performance that can the suggested system can gain. Meanwhile, the highest True Negative Rate is 0.87 as the lowest TPR is 0.80. The ROC curve as shown in 5.4 explained an increasing detection rate using NIHA. WEKA data mining tool is used to evaluate the suggested new data which has enhanced the detection process.

5.9 Summary

In this chapter, an evaluation and cross validation process of the model HeuBNet6 have been performed. Furthermore, dynamic association analysis was

considered for the results given by NIHA in the empirical period and overall framework testing. Critical analysis of the integrated MNBC module has been performed as well. Additionally, technical issues such as design and implementation has been discussed. A comparison between HeuBNet6 and current techniques with regards to the compatibility and flexibility in business needs and network security has been discussed. The model's drawbacks have been discussed. Moreover, the barriers in performing such analysis on live detection have been discussed. Static analysis resulted in considerable precision rate and confusion metrics level for the attributes and its subset values.

Three outcome reports, namely the statistical, analytical and evaluational reports were discussed in this chapter, and the liability level of the model with regards to its feasibility against such live attacks in network security were explained. The experiments results showed the accuracy of the detection methodology used in HeuBNet6 and elaborated the extensibility of detecting unknown and new security attacks against IPV6 protocol. Our system has detected and analysed more than 20 possible covert channels in IPv6 causing various security threats against legitimate targets. The frequent extensibility of such attacks depended on the attacker's knowledge about the system, inherited protocol vulnerability in architecture design and the users' behavioural patterns. All these elements are vital for network engineers and cyber security experts to tackle and mitigate these types of attacks.

Chapter 6

Conclusions and Future Work

In this chapter, conclusions of the undertaken work in the thesis will be highlighted in addition to the major achieved contributions. Some significant recommendations and suggestions will be presented along with the future work directions.

6.1 Conclusions

Network security is an inevitable demand and a major concern for most governments, agencies and professional companies. This is due to their huge involvement in the internet and networking business transactions. Hence, as mentioned in Section 2.7 there is a need for new approaches with new directions to implement different methodologies not only to mitigate security threats but also to eliminate these vulnerabilities of TCP/IPv6. The heuristic intelligent algorithm is one of the most powerful techniques used in searching and making decisions. This approach outcomes with the best optimal results in many areas, but it has not been used in tackling, detecting and eliminating covert channels in IPv6 so far. Due to various and different approaches in network security, the need for this technique arose in our preliminary investigations and analysis in security vulnerabilities as discussed in Sections 1.1 and 1.3. Additionally, implementing Advanced Machine Learning algorithm in the same domain also was a vital option to develop our framework and challenge such types of hidden information carrier over the net.

Our Network Intelligent Heuristic Algorithm (NIHA) and the enhanced Multinomial Naïve Bayes Classifier (MNBC) were gathered as "HeuBNet6" to represent a new type of hybridization security framework. This novel adaptive framework fights against covert channels within IPv6. It works as an advanced Intrusion Detection and Prevention System (IDPS) for IPv6. Furthermore, it uses supervised learning technique objectively to mitigate security threats and attacks against IPv6 and ICMPv6 protocols. This hybrid approach is adding a different and a new way of thinking towards finding solutions to one of the most sophisticated attacks against privacy and security policies in the world [74, 181].

The suggested classification model showed a high accuracy rate and improved the performance of Multinomial Naïv Bayes Algorithm (MNBA). This development was elaborated through coupling MNBA with feature selection techniques in this dual (multi-layer) security (multi-threaded) mechanism. IPv6 has some serious security vulnerabilities despite of its security improvements and advanced functionalities in comparison to IPv4. Covert channel is categorized as one of the most effective stealthy security threats against any organization's sensitive (classified) data or information assets; eventually, it violates network security policies and privacy rights of the users.[5, 17, 25, 26].

Generally, as mentioned in Chapters 3 and 4, an advanced feature selection technique has been used in the latter part of the model. C4.5 decision trees with Information Gain technique was developed and implemented successfully to achieve higher accuracy rate and lower possible faults. This method has significant development phases in data pruning and presenting a clear format of the most informative tuples belonging to the main attributes. It functioned coherently in the process development to classify and detect covert channels in IPv6. Moreover, this process was performed preliminary to analyse the live data type of each captured packet. These packets carried substances of various attack instances causing various damage to classified information assets in vulnerable machines. This technique was implemented to categorize and optimize values held in each attack instance with its subset values. Metrics and measurements used to perform the aimed objectives were according to the variation of the most standard values set by RFC's and IANA. The suggested algorithm depicted most abnormal behaviour of the selected attributes in the header field of IPv6. These

attributes could be misused and reversed against legitimate targets.

In this thesis, the development of the experimental steps was conducted to reduce the probabilistic stimulation, which developed the process in such a way to achieve higher accuracy in detecting common sophisticated attacks, which manipulated protocols' headers and payloads vulnerabilities. Furthermore, it led to a successful classification process, then eventually, achieved lower false negative rate (FNR) and higher true positive rate (TPR).

The reason behind this optimal level of performance, as shown in test the phases in Section 4.5 and the analysis in Chapter 5, was that a different approach has been deployed to deal with diverted and transformed complex data format within the preprocessing data module. Significantly, this progress was used to reduce data entropy and noise in both training and test datasets. However, in covert channel analysis, it was found that the original values of IPv6 protocol header fields were not always recognized by NIHA which possibly indicated to contain certain amount of entropy. This vulnerability has been exploited by the attacker to inject malware or any type of data aiming to the receiver then to create serious damages.

This step particularly initialized pure data pruning process and significant compatible data format (attack instances) outcome as shown in Chapter 4. Consequently, the module was able to create original primary data which was considered to be the essence of the new approach and implementation. This was due to the lack of covert channel instances attributes (data type format) merely for IPv6. These formats were analysed and depicted to enable and direct the process to expose hidden channel features which were selected in the primary dataset of the IPv6 and its attacks in captured packets.

In this thesis, it was managed to depict most of usability chances to abuse IPv6 header fields. Furthermore, more than 55 test cases were run in a controlled network lab environment, and approximate 70% decent and clear experimented cases were chosen and validated as shown in Section 4.5.

The following research questions have been answered in the thesis:

1. What is the feasibility of a new prevention model to detect and eliminate covert channel attacks in IPv6? It is argued that a prevention model is more likely possible to be designed and created in order to perform detection and mitigation of covert channels. We have achieved the planned primary objective of the research question. Furthermore, it was confirmed that as far as there are new technologies every day, there would be a need for new prevention models. This issue was discussed in Sections 2.2 and 2.3.
2. What effective countermeasure methods can be used to mitigate security implications and the associated risks in IPv6? This question was discussed in Sections 2.3 and 3.3. The need for advanced countermeasures through investigating the traditional techniques used in current and previous countermeasures has been explained. Moreover, the security threats impact created by attackers on network security policies and privacy was also explained. Furthermore, a justification of the new findings via practical evidence and evaluation of the suggested new model was presented in comparison to other different models to mitigate such vulnerabilities in IPv6.

Furthermore, the sub questions also have been answered in this thesis as follows:

- (a) Is it possible to develop a classifier to categorize an arbitrarily sized database of labelled attack instances and test the conditional independence status of the attributes? A classifier has been developed to organize and categorize the data which was created by NIHA. In Section 3.6, an elaboration about the classifier MNBC is given. Meanwhile, MNBC is processed as a second security module detecting the covert data and its attack instances.
- (b) Can a supervised Machine Learning technique be developed to create a knowledge based framework in order to detect and classify covert channels in IPv6? This sub question also has been answered through the development of the project. An automated machine learning technique

is used in the suggested knowledge based framework. Furthermore, it is a novel method used in detecting and classifying covert channels in IPv6. Section 3.5 explains in detail the framework steps and phases.

- (c) Is it possible to optimize the accuracy of covert channels classification using advanced feature selection technique? The accuracy was invoked through customizing the subset values of each attack attribute. NIHA module was used to create the novel attack instances; meanwhile, MNBC module was used to customize the values using enhanced feature selection technique such as C4.5. Section 3.5 elaborates the technical implementation of the module in the framework.

6.2 Summary of the Contributions

This project has contributed to propose a new approach different from current approaches to solve the similar problem which many researchers have attempted so far. It analyses, classifies and detects the most complicated IPv6 protocol security vulnerabilities using advanced intelligent heuristic algorithms [17, 25]. This technique functioned as a primary model for data processing. Due to the lack of covert channels benchmark database, provisional samples have been created through simulations of various local attacks on a controlled separated LAN topology. The scope of the project was narrowed according to ethical rules restrictions and Acts as explained in Section 6.3. The primary contribution of this research is that to the best of our knowledge, this is the first effort that suggests a multi-threaded security system for IPv6 to mitigate such complicated security threats created by covert channels attacks. Furthermore, to the best of our knowledge, in a part of the attempts performed [17, 25, 26], there is no sufficient data (known and unknown attack instances) of using covert channels in IPv6 similar to what has been achieved in this project. In this project, primary data was developed which consisted of nearly 6 millions multi instances data types. These data types can be used and manipulated for future research in this domain for ongoing development research basis.

The major contributions presented in this thesis are illustrated as follows.

6.2.1 A New Structure for Security Detection Model

A multi-threaded security structure is designed for IPv6 to detect covert channels (information carrier) attacks and threats. The flexibility of this structure is offering a new attempt to deal with high speed traffic flow and throughput. This part was presented in Section 3.5.

The model uses advanced data mining techniques as a supervised machine learning algorithm in an enhanced mode of Multinomial Naïve Bayes Classifier (MNBC) coupled with a novel heuristic analysis approach. This model is an adaptive security approach which can be used as a modern paradigm for threat detection and classification in future accumulative research. This part of contribution was explained in Section 3.6.

6.2.2 A Novel Intelligent Heuristic Algorithm

Heuristics are mental short cut or "rule of thumbs" that give some guidance on how to do a tasks, but it does not guarantee solutions consistently [182]. Heuristics may be used to determine the specific rules for solution generation. The research contributes a novel Network Intelligent Heuristic Algorithm (NIHA) as an essential security system model to analyse, assess and create covert channel characteristics in IPv6 and ICMPv6 packets. This will optimise the weight and value of each attack instance. This part of the contribution was elaborated in Section 3.7.

The proposed algorithm NIHA is the core element of the security detection model as it provides the means to specify and manage detection policies and specifications that are used to detect covert channels in IPv6.

6.2.3 A New Primary Dataset for Covert Channel Attacks in IPv6

The creation of new primary data through the use of NIHA is one of the distinctive contribution in this project and it is a leap into ground breaking areas of IPv6 covert channels¹. New attack instances are created as an outcome of the empirical research progress. The primary data for the initial development of the project

¹Test scenarios and data are available at <https://ultimatecyber.github.io/heubnet6/>

are potentially impressive to the testing and evaluation phases. Furthermore, this data can be used for future research and similar models evaluation particularly in TCP/IPv6 security problem domain. This part of contribution was presented in Section 3.7.

6.2.4 A New Shared Knowledge Framework

A new shared knowledge framework is presented and implemented in this research supported by the developed security model and a new sample research dataset produced by NIHA. The knowledge base can be redeployed at all critical points of an organisation's network topology. It is envisaged that further research emanating from this work will produce the amalgamation of newly discovered attacks due to changes of IPv6 specifications acquired through the wider research community.

6.3 Limitations of The Project

Since the study of covert channels started to gain momentum in the security community about two decades ago, numerous diverse ideas have been formulated to confront and counter this problem. To detect steganography or cryptography, advanced in-depth analysis encryption application is needed. Meanwhile, metaferography (covert channel) detection may vary using different sources to obtain data then to manipulate specific techniques to analyse the data. Apparently, most of the security systems classify data either by misuse detection or behaviour (anomaly) detection. Each approach presents its relevant metrics with certain limitations. It is unlikely to expect that an IDS or IPS can be capable to eliminate every single channel used for data transfer. However, there is no perfect detection system. This is not possible at all due to the complexity, incomplete implementation, and the unpredictable state of the fast advanced system evolution.

A single IPv6 IDS or IPv6 covert channel detector can strive to narrow the success opportunities for attackers by minimizing the probabilities of large attack classes. The coordinated implementation of multi-thread security system promises and allows broader confidence in the given results. Additionally, it improves the

coverage of hidden communication channels detection process, making it a critical component for different comprehensive security system architecture.

6.3.1 Ethical Consideration

It is impossible to create a research free of delimitations aspects, so as this research which was limited by the ethical rules considerations represented in Data Protection Act 1998. The project empirical stage testing would harm and violate public and private network policy and privacy if operated; however, the rules and regulations do not allow penetration testing or "hacking" in an academic institution like Nottingham Trent University. Therefore, a controlled network topology design was created using GNS3 network simulation tool [164]. This tool was installed on Oracle VM Virtual Box Graphical User Interface Version 5.0.16 r105871. Eventually, the project would have become more effective if we could have performed some real attacks online against our virtual targeted servers.

6.4 Future Work

The investigation about covert channel in IPv6 is new and needs more qualitative studies offering different opportunities for future research. Most of the security issues studied and investigated in this thesis carried specific path for more inquiry to explore new findings. However, IPv6 has an ongoing design aspects development which allows further research.

The flexibility in the protocol design facilitates attackers' job to in /ex-filtrate any type of data from legitimate targets. Supposedly, this should draw attention of the protocol designers to reconsider the protocol architecture to the existence of covert channels and think about the mitigation options. Obviously, IPsec has made IPv6 more secure than IPv4 in terms of the obligation in activating it. Meanwhile, the later protocol application is more vulnerable when IPsec is deployed for packet integrity check with gaining complete knowledge about the IPsec security. Thus, covert channels could not be defeated as far as the security information is disclosed. Therefore, the inherited vulnerabilities in protocol design have contributed into the existence of metaferography (covert channels), hence the complete elimination is

impossible now and in the future.

Despite the fact that it is inevitable to have a prerequisite knowledge when countering covert channels, they are more likely available in the upper layer protocols such as: TCP, UDP, HTTP etc., and need to be addressed in order to perform more investigations about the nature of such unknown anomaly attacks. Accordingly, after achieving the main and sub objectives of this thesis, other questions have been raised which inspire future research such as:

1. Can different security frameworks be used to test the produced primary data in order to improve the flexibility and compatibility with other security frameworks to reach standard benchmark level?
2. What are the chances to achieve faster detection and better performance to solve similar security issues for different protocols such as TCP and UDP?
3. What are the security implications on business needs and information management in IPv6 network?
4. Can covert channels use embedded steganography in IPv6 network?

These questions leave potential indications and directions for future work and further planning. The motivation is to examine the weighting and ranking of different covert channels in different security protocols to defeat cyber security threats and attacks against legitimate targets and institutions. Furthermore, it is important to develop feature selection algorithms to obtain the most optimal accuracy using the tuples in the primary dataset of the IPv6. In addition to this, it is desirable to design new and compatible techniques which can process data tree pruning via using different and advanced feature selection methods. Although fuzzy genetic technique has been investigated and tested [21] in the cross validation phase, it gave different accuracy rate in comparison to the given results in this thesis. HeuBNet6 can be developed into other directions by extending the compatibility of the measures investigated into broader extensibility such as time covert channel which will enhance and enrich the research domain in the future.

Appendices

A.1 HeuBNet6 and ICMPv6 with TypeCode Algorithm

Algorithm 12 HeuBNet6_ICMPv6_TypeCode_Anomaly_Detection_Algorithm

INPUT: $IPv6_PKT$, $Class_c$

OUTPUT: Nominal Value tcd_{nom} , Classification $Class_c$.

```
1:  $IPv6\_HDR \leftarrow GET\_IPV6\_HDR(IPv6\_PKT)$ 
2:  $ICMPv6\_HDR \leftarrow GET\_ICMPV6\_HDR(IPv6\_PKT)$ 
3: function CHECK_TYPE_CODE_PAIR( $Thread_i$ ;  $Queue$ )
4:   if  $ICMPv6\_HDR_{type} = ICMP6\_ECHO\_REQUEST$  and  $ICMPv6\_HDR_{code} \neq$ 
       $ALLOWED\_ECHO\_REQUEST\_CODE$  then{
5:      $Class_c \leftarrow ANOMALY$ 
6:     go to DST_RETURN_RESULT
7:   }
8:   end if
9:   if  $ICMPv6\_HDR_{type} = ICMP6\_ECHO\_REPLY$  and  $ICMPv6\_HDR_{code} \neq$ 
       $ALLOWED\_ECHO\_REPLY\_CODE$  then{
10:     $Class_c \leftarrow ANOMALY$ 
11:    go to DST_RETURN_RESULT
12:   }
13:   end if
14:   if  $ICMPv6\_HDR_{type} = ICMP6\_DST\_UNREACH$  and  $ICMPv6\_HDR_{code} \notin$ 
       $ALLOWED\_DST\_UNREACH\_CODES$  then
15:      $Class_c \leftarrow ANOMALY$ 
```

```

16:     go to DST_RETURN_RESULT
17:  else
18:     if ICMPv6_HDRcode = 0 then
19:         dst_exists                                     ←
CHECK_DST_ADDR_IN_TOPOLOGY_CORPUS(ICMPv6_HDRdst_addr)
20:         if dst_exists = FALSE then
21:             Classc ← ANOMALY
22:             go to DST_RETURN_RESULT
23:         end if
24:     end if
25:     if ICMPv6_HDRcode = 1 then
26:
access_denied ← CHECK_DST_ADDR_ACCESSIBLE(ICMPv6_HDRdst_addr)
27:         if access_denied = TRUE then
28:             Classc ← ANOMALY
29:             go to DST_RETURN_RESULT
30:         end if
31:     end if
32:     if ICMPv6_HDRcode = 2 then
33:         is_assigned ← CHECK_DST_ADDR_ASSIGNED(ICMPv6_
HDRdst_addr)
34:         if is_assigned = FALSE then
35:             Classc ← ANOMALY
36:             go to DST_RETURN_RESULT
37:         end if
38:     end if
39:     if ICMPv6_HDRcode = 3 then
40:
addr_unreachable ← CHECK_DST_ADDR_REACH(ICMPv6_HDRdst_addr)
41:         if addr_reachable = FALSE then
42:             Classc ← ANOMALY
43:             go to DST_RETURN_RESULT
44:         end if

```

```

45:     end if
46:     if ICMPv6_HDRcode = 4 then
47:         port_unreachable ←
CHECK_DST_ADDR_PORT_REACH(ICMPv6_HDRdst_addr)
48:         if port_reachable = FALSE then
49:             Classc ← ANOMALY
50:             go to DST_RETURN_RESULT
51:         end if
52:     end if
53: end if
54: DST_RETURN_RESULT:
    ▷ String Concatenation
55:     tcdnom ← Type + ICMPv6_HDRty + _Code + ICMPv6_HDRcd
56:     return tcdnom, class
57: end function
58:
59:
60: function DROP_BLOCKED_MESSAGE_TYPES(Pkt, Result)
61:     DUM := 1                ▷ Destination Unreachable Messages
62:     PTBM := 2              ▷ Packet Too Big Messages
63:     TEM := 3               ▷ Time Exceeded Messages
64:     PPM := 4               ▷ Parameter Problem Messages
65:     MLD := {130,131,132,143} ▷ Multicast Listener Discovery
66:     NET := {...}          ▷ Local Net Address Pool
67:     if IPv6_PKTICMPv6_HDRtype = {DUM or PTBM or TEM or PPM} then
68:
    ▷ =====
    ▷ verify exceeding MTU
    ▷ =====
69:     if IPv6_PKTICMPv6_HDRtype ∈
PTBM and PACKET_EXCEEDS_MTU(IPv6_PKT) then
70:         Classc ← ANOMALY
71:     end if

```

```

72:
    ▷ =====
    ▷ tem -0 - Hop limit exceeded in transit 1 - Fragment reassembly time exceeded
    ▷ =====
73:     if IPv6_PKTICMPv6_HDRtype ∈
    TEM and HOP_LIMIT_EXCEEDED(IPv6_PKThl) then
74:         Classc ← ANOMALY
75:     end if
76:     if IPv6_PKTICMPv6_HDRtype ∈
    TEM and HOP_LIMIT_EXCEEDED(IPv6_PKThl) then
77:         Classc ← ANOMALY
78:     end if
79:
    ▷ =====
    ▷ ppm - 0 - Erroneous header field encountered 1 - Unrecognized next header
    type encountered 2 - Unrecognized IPv6 option encountered
    ▷ =====
80:     if IPv6_PKTICMPv6_HDRtype ∈
    PPM and UNKOWN_HEADER_FIELD_FOUND(IPv6_PKT) then
81:         Classc ← ANOMALY
82:     end if
83:     if IPv6_PKTICMPv6_HDRtype ∈
    PPM and UNKOWN_NEXT_HEADER_TYPE_FOUND(IPv6_PKT) then
84:         Classc ← ANOMALY
85:     end if
86:     if IPv6_PKTICMPv6_HDRtype ∈
    PPM and UNKOWN_IPV6_OPTION(IPv6_PKT) then
87:         Classc ← ANOMALY
88:     end if
89:     Resultstate ← DROP
90: end if
    ▷ Check MLD errors
91:     if PktICMPv6_HdrType ∈ MLD and {PktIPv6_HdrSrc_addr ∉

```

```

    NET or  $Pkt_{IPv6\_Hdr\_Dst\_addr} \notin NET$  } then
92:      $Result_{state} \leftarrow DROP$ 
93:   end if
94:   return  $Result$ 
95: end function
96:
97: function ALLOW_PERMITTED_MESSAGE_TYPES( $Pkt, Result$ )
98:    $UEM := \{5-99, 102-126\}$            ▷ Unallocated Error Messages
99:    $UIM := \{155-199, 202-254\}$        ▷ Unallocated Information Messages
100:   $EM := \{100, 101, 200, 201\}$        ▷ Experimental Messages
101:   $ETN := \{127, 255\}$                ▷ Extension Type Numbers
102:
103:  if  $Pkt_{ICMPv6\_HdrType} \in \{UEM \cup UIM \cup EM \cup ETN\}$  then
104:     $Result_{state} \leftarrow ALLOW$ 
105:  end if
106:  return  $Result$ 
107: end function

```

– End of Type-Code Algorithm –

A.2 Investigated Request For Comments Protocols

List of Main RFCs Investigated and Examined for This Project

1. Internet Protocol, Version 6 (IPv6) Specification - RFC 2460
2. Router Alert - RFC 2711
3. IPv6 jumbogram - RFC 2675
4. Tunnel Encapsulation Limit - RFC 2473
5. IPv6 Flow Label Specification - RFC 3697
6. IP Mobility Home Address - RFC 6275
7. IP Mobility Home Address - RFC 6275
8. Using the IPv6 Flow Label for Equal Cost Multipath Routing and Link Aggregation in Tunnels - 6438
9. IPv6 Flow Label Specification - RFC 6437

A.3 Flow Label Manager "Corpus"

Flow Label Manager "Corpus"

Table 1: Corpus of Flow Label Manager

id,i,0	src_addr,s,1	dst_addr,s,2	proto,i,1	f_label,i,2	alert_mode
1001	2001::7	2000::3	58	99	alert,v,1,\$msg1\$dst_addr\$dst_addr_port\$Id
1002	2001::7	2000::3	58	101	alert,v,1,\$msg1\$dst_addr\$dst_addr_port\$Id
1003	2001::7	2000::3	58	-1	alert,v,1,\$msg1\$dst_addr\$dst_addr_port\$Id
1004	2001::7	2000::14	58	23	alert,v,1,\$msg1\$dst_addr\$dst_addr_port\$Id
1005	2001::7	2000::14	58	1000	alert,v,1,\$msg1\$dst_addr\$dst_addr_port\$Id
1006	2001::7	2000::17	58	-1	alert,v,1,\$msg1\$dst_addr\$dst_addr_port\$Id
1007	2001::7	2000::17	58	12	alert,v,1,\$msg1\$dst_addr\$dst_addr_port\$Id
1008	2001::7	2000::17	58	166	alert,v,1,\$msg1\$dst_addr\$dst_addr_port\$Id
1009	2001::7	2000::17	58	500	alert,v,1,\$msg1\$dst_addr\$dst_addr_port\$Id
1010	2001::13	2000::3	58	99	alert,v,1,\$msg1\$dst_addr\$dst_addr_port\$Id
1011	2001::13	2000::3	58	653	alert,v,1,\$msg1\$dst_addr\$dst_addr_port\$Id
1012	2001::13	2000::3	58	742	alert,v,1,\$msg1\$dst_addr\$dst_addr_port\$Id
1013	2001::13	2000::14	58	7222	alert,v,1,\$msg1\$dst_addr\$dst_addr_port\$Id
1014	2001::13	2000::14	58	33	alert,v,1,\$msg1\$dst_addr\$dst_addr_port\$Id
1015	2001::13	2000::17	58	52	alert,v,1,\$msg1\$dst_addr\$dst_addr_port\$Id
1016	2001::13	2000::17	58	57	alert,v,1,\$msg1\$dst_addr\$dst_addr_port\$Id
1017	2001::15	2000::3	58	-1	alert,v,1,\$msg1\$dst_addr\$dst_addr_port\$Id
1018	2001::15	2000::14	58	-1	alert,v,1,\$msg1\$dst_addr\$dst_addr_port\$Id
1019	2001::15	2000::17	58	-1	alert,v,1,\$msg1\$dst_addr\$dst_addr_port\$Id

A.4 Sample of Training Dataset

Table 2: HeuBNet6 Training Dataset Sample

Class	TF	FL	PL	NH	SA	DA	TY	CD	PY
anomaly	1	99	47	60	2001::7	2000::3	0	0	payload
anomaly	1	99	47	43	2001::7	2000::3	0	0	payload
anomaly	1	99	47	44	2001::7	2000::3	0	0	payload
anomaly	1	99	47	51	2001::7	2000::3	0	0	payload
anomaly	1	99	47	50	2001::7	2000::3	0	0	payload
anomaly	1	99	47	135	2001::7	2000::3	0	0	payload
anomaly	1	99	47	59	2001::7	2000::3	0	0	payload
anomaly	1	99	47	6	2001::7	2000::3	0	0	payload
normal	1	99	47	58	2001::7	2000::3	0	0	payload
anomaly	1	99	47	60	2001::7	2000::3	0	1	payload
anomaly	1	99	47	43	2001::7	2000::3	0	1	payload
anomaly	1	99	47	44	2001::7	2000::3	0	1	payload
anomaly	1	99	47	51	2001::7	2000::3	0	1	payload
anomaly	1	99	47	50	2001::7	2000::3	0	1	payload
anomaly	1	99	47	135	2001::7	2000::3	0	1	payload
anomaly	1	99	47	59	2001::7	2000::3	0	1	payload
anomaly	1	99	47	6	2001::7	2000::3	0	1	payload
normal	1	99	47	58	2001::7	2000::3	0	1	payload
anomaly	1	99	47	60	2001::7	2000::3	0	2	payload
anomaly	1	99	47	43	2001::7	2000::3	0	2	payload
anomaly	1	99	47	44	2001::7	2000::3	0	2	payload
anomaly	1	99	47	51	2001::7	2000::3	0	2	payload
anomaly	1	99	47	50	2001::7	2000::3	0	2	payload
anomaly	1	99	47	135	2001::7	2000::3	0	2	payload
anomaly	1	99	47	59	2001::7	2000::3	0	2	payload
anomaly	1	99	47	6	2001::7	2000::3	0	2	payload
normal	1	99	47	58	2001::7	2000::3	0	2	payload
anomaly	1	99	47	60	2001::7	2000::3	0	3	payload
anomaly	1	99	47	43	2001::7	2000::3	0	3	payload
anomaly	1	99	47	44	2001::7	2000::3	0	3	payload
anomaly	1	99	47	51	2001::7	2000::3	0	3	payload
anomaly	1	99	47	50	2001::7	2000::3	0	3	payload
anomaly	1	99	47	135	2001::7	2000::3	0	3	payload
anomaly	1	99	47	59	2001::7	2000::3	0	3	payload
anomaly	1	99	47	6	2001::7	2000::3	0	3	payload
normal	1	99	47	58	2001::7	2000::3	0	3	payload
anomaly	1	99	47	60	2001::7	2000::3	0	4	payload
anomaly	1	99	47	43	2001::7	2000::3	0	4	payload
anomaly	1	99	47	44	2001::7	2000::3	0	4	payload
anomaly	1	99	47	51	2001::7	2000::3	0	4	payload

Table 3: HeuBNet6 Training Dataset Sample

Class	TF	FL	PL	NH	SA	DA	TY	CD	PY
anomaly	1	99	47	135	2001::7	2000::3	0	12	payload
anomaly	1	99	47	59	2001::7	2000::3	0	12	payload
anomaly	1	99	47	6	2001::7	2000::3	0	12	payload
normal	1	99	47	58	2001::7	2000::3	0	12	payload
anomaly	1	99	47	60	2001::7	2000::3	0	13	payload
anomaly	1	99	47	43	2001::7	2000::3	0	13	payload
anomaly	1	99	47	44	2001::7	2000::3	0	13	payload
anomaly	1	99	47	51	2001::7	2000::3	0	13	payload
anomaly	1	99	47	50	2001::7	2000::3	0	13	payload
anomaly	1	99	47	135	2001::7	2000::3	0	13	payload
anomaly	1	99	47	59	2001::7	2000::3	0	13	payload
anomaly	1	99	47	6	2001::7	2000::3	0	13	payload
normal	1	99	47	58	2001::7	2000::3	0	13	payload
anomaly	1	99	47	60	2001::7	2000::3	0	14	payload
anomaly	1	99	47	43	2001::7	2000::3	0	14	payload
anomaly	1	99	47	44	2001::7	2000::3	0	14	payload
anomaly	1	99	47	51	2001::7	2000::3	0	14	payload
anomaly	1	99	47	50	2001::7	2000::3	0	14	payload
anomaly	1	99	47	135	2001::7	2000::3	0	14	payload
anomaly	1	99	47	59	2001::7	2000::3	0	14	payload
anomaly	1	99	47	6	2001::7	2000::3	0	14	payload
normal	1	99	47	58	2001::7	2000::3	0	14	payload
anomaly	1	99	47	60	2001::7	2000::3	0	15	payload
anomaly	1	99	47	43	2001::7	2000::3	0	15	payload
anomaly	1	99	47	44	2001::7	2000::3	0	15	payload
anomaly	1	99	47	51	2001::7	2000::3	0	15	payload
anomaly	1	99	47	50	2001::7	2000::3	0	15	payload
anomaly	1	99	47	135	2001::7	2000::3	0	15	payload
anomaly	1	99	47	59	2001::7	2000::3	0	15	payload
anomaly	1	99	47	6	2001::7	2000::3	0	15	payload
normal	1	99	47	58	2001::7	2000::3	0	15	payload
anomaly	1	99	47	60	2001::7	2000::3	0	16	payload
anomaly	1	99	47	43	2001::7	2000::3	0	16	payload
anomaly	1	99	47	44	2001::7	2000::3	0	16	payload
anomaly	1	99	47	51	2001::7	2000::3	0	16	payload
anomaly	1	99	47	50	2001::7	2000::3	0	16	payload
anomaly	1	99	47	135	2001::7	2000::3	0	16	payload
anomaly	1	99	47	59	2001::7	2000::3	0	16	payload
anomaly	1	99	47	6	2001::7	2000::3	0	16	payload
normal	1	99	47	58	2001::7	2000::3	0	16	payload

Bibliography

- [1] A Behrouz Forouzan, *Data Communications & Networking (sie)*, Tata McGraw-Hill Education, 2006. [xii](#), [15](#), [18](#), [22](#), [59](#)
- [2] Silvia Hagen, *IPv6 essentials*, " O'Reilly Media, Inc.", 2006. [xii](#), [xiv](#), [17](#), [18](#), [19](#), [23](#), [24](#), [27](#), [36](#), [37](#), [48](#)
- [3] Abdur Rahim Choudhary, "In-depth analysis of ipv6 security posture", in *2009 5th International Conference on Collaborative Computing: Networking, Applications and Worksharing*, 2009. [xii](#), [26](#), [27](#), [48](#)
- [4] Kaining Lu, Zehua Chen, Zhigang Jin, and Jichang Guo, "An real-time intrusion detection system using sequences of system call", in *Electrical and Computer Engineering, 2003. IEEE CCECE 2003. Canadian Conference on*. IEEE, 2003, vol. 2, pp. 789–792. [xii](#), [34](#)
- [5] Norka B Lucena, Grzegorz Lewandowski, and Steve J Chapin, "Covert channels in ipv6", in *Privacy Enhancing Technologies*. Springer, 2006, pp. 147–166. [xii](#), [xiv](#), [3](#), [7](#), [13](#), [15](#), [16](#), [21](#), [23](#), [25](#), [26](#), [27](#), [28](#), [31](#), [32](#), [33](#), [34](#), [52](#), [53](#), [54](#), [55](#), [66](#), [165](#), [174](#)
- [6] Jayveer Singh and Manisha J Nene, "A survey on machine learning techniques for intrusion detection systems", *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 2, no. 11, pp. 4349–4355, 2013. [xii](#), [43](#), [46](#), [152](#)
- [7] Sebastian Zander, Grenville Armitage, and Philip Branch, "A survey of covert channels and countermeasures in computer network protocols",

- Communications Surveys & Tutorials, IEEE*, vol. 9, no. 3, pp. 44–57, 2007. [xii](#), [3](#), [14](#), [26](#), [33](#), [51](#)
- [8] ZQ John Lu, “The elements of statistical learning: data mining, inference, and prediction”, *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, vol. 173, no. 3, pp. 693–694, 2010. [xiii](#), [149](#), [150](#)
- [9] Scott Hogg and Eric Vyncke, *IPv6 security*, Pearson Education, 2008. [xiv](#), [17](#), [20](#), [21](#), [22](#), [23](#), [27](#), [31](#), [37](#)
- [10] Norka B Lucena, Grzegorz Lew, and Steve J Chapin, “Eliminating covert channels in ipv6 with network-aware active wardens”. [xiv](#), [34](#), [54](#)
- [11] Grzegorz Lewandowski, Norka B Lucena, and Steve J Chapin, “Analyzing network-aware active wardens in ipv6”, in *Information Hiding*. Springer, 2007, pp. 58–77. [xiv](#), [13](#), [35](#), [48](#), [52](#), [54](#)
- [12] Taeshik Sohn, J. Seo, and Jongsub Moon, “A study on the covert channel detection of tcp/ip header using support vector machine”, in *ICICS*. 2003, pp. 313–324, Springer. [1](#), [14](#), [17](#), [30](#), [37](#), [70](#)
- [13] Butler W Lampson, “A note on the confinement problem”, *Communications of the ACM*, vol. 16, no. 10, pp. 613–615, 1973. [1](#)
- [14] Steven J. Murdoch and Stephen Lewis, “Embedding covert channels into tcp/ip”, in *Information hiding*. 2005, pp. 247–261, Springer. [2](#), [14](#), [16](#)
- [15] Neil F Johnson, Zoran Duric, and Sushil Jajodia, *Information Hiding: Steganography and Watermarking-Attacks and Countermeasures: Steganography and Watermarking: Attacks and Countermeasures*, vol. 1, Springer Science & Business Media, 2001. [2](#), [14](#), [165](#)
- [16] Stephen E Deering, “Internet protocol, version 6 (ipv6) specification”, 1998. [2](#), [18](#), [21](#)
- [17] Grzegorz Lewandowski, Norka B. Lucena, and Steve J. Chapin, “Analyzing network-aware active wardens in ipv6”, in *Information Hiding*. 2007, pp. 58–77, Springer. [2](#), [7](#), [15](#), [16](#), [23](#), [33](#), [39](#), [48](#), [52](#), [53](#), [54](#), [55](#), [132](#), [174](#), [177](#)

- [18] Wesley K Clark and Peter L Levin, “Securing the information highway: how to enhance the united states’ electronic defenses”, *Foreign affairs*, pp. 2–10, 2009. [2](#)
- [19] Matthew V Mahoney and Philip K Chan, “An analysis of the 1999 darpa/lincoln laboratory evaluation data for network anomaly detection”, in *Recent Advances in Intrusion Detection*. Springer, 2003, pp. 220–237. [3](#)
- [20] Supriyanto, Raja Kumar Murugesan, and Sureswaran Ramadass, “Review on ipv6 security vulnerability issues and mitigation methods”, *International Journal of Network Security & Its Applications*, vol. 4, no. 6, pp. 173, 2012. [3](#), [14](#), [15](#), [25](#), [26](#), [28](#), [29](#), [49](#), [52](#)
- [21] Abdulrahman Salih, Xiaoqi Ma, and Evtim Peytchev, “Implementation of hybrid artificial intelligence technique to detect covert channels attack in new generation internet protocol ipv6”, in *Leadership, Innovation and Entrepreneurship as Driving Forces of the Global Economy*, pp. 173–190. Springer, 2017. [3](#), [39](#), [41](#), [42](#), [46](#), [166](#), [168](#), [181](#)
- [22] Amirhossein Moravejosharieh, Hero Modares, and Rosli Salleh, “Overview of mobile ipv6 security”, in *Intelligent Systems, Modelling and Simulation (ISMS), 2012 Third International Conference on*. IEEE, 2012, pp. 584–587. [5](#)
- [23] Kamran Ahsan and Deepa Kundur, “Practical data hiding in tcp/ip”, in *Proc. Workshop on Multimedia Security at ACM Multimedia*, 2002, vol. 2. [5](#), [14](#), [15](#), [25](#)
- [24] Richard Savacool, “Firewall resistance to metaferography in network communications”, 2010. [5](#), [13](#), [168](#)
- [25] Steffen Wendzel, Benjamin Kahler, and Thomas Rist, “Covert channels and their prevention in building automation protocols: A prototype exemplified using bacnet”, in *Green Computing and Communications (GreenCom), 2012 IEEE International Conference on*. IEEE, 2012, pp. 731–736. [7](#), [33](#), [174](#), [177](#)

- [26] Sebastian Zander, Grenville Armitage, and Philip Branch, “A survey of covert channels and countermeasures in computer network protocols”, *Communications Surveys & Tutorials, IEEE*, vol. 9, no. 3, pp. 44–57, 2007. [7](#), [23](#), [25](#), [27](#), [31](#), [43](#), [48](#), [50](#), [54](#), [174](#), [177](#)
- [27] William J Buchanan and David Llamas, “Covert channel analysis and detection with reverse proxy servers using microsoft windows”, in *Proceedings of the 3rd European conference on information warfare and security*. Academic Conferences Limited, 2004, pp. 31–40. [7](#), [56](#)
- [28] David Lorge Parnas, “On the criteria to be used in decomposing systems into modules”, *Communications of the ACM*, vol. 15, no. 12, pp. 1053–1058, 1972. [12](#)
- [29] Ross J. Anderson and Fabien AP Petitcolas, “Information hiding”, *Signal Processing*, vol. 80, pp. 2067–2070, 1996. [12](#), [13](#)
- [30] Steffen Wendzel and Jörg Keller, “Hidden and under control”, *annals of telecommunications-Annales des télécommunications*, vol. 69, no. 7-8, pp. 417–430, 2014. [12](#), [49](#)
- [31] Birgit Pfitzmann, “Information hiding terminology-results of an informal plenary meeting and additional proposals”, in *Proceedings of the First International Workshop on Information Hiding*. Springer-Verlag, 1996, pp. 347–350. [12](#), [13](#)
- [32] Neil F. Johnson and Sushil Jajodia, “Steganalysis: The investigation of hidden information”, in *Information Technology Conference, 1998. IEEE*. 1998, pp. 113–116, IEEE. [12](#)
- [33] Sebastian Zander, Grenville Armitage, and Philip Branch, “Covert channels in the ip time to live field”, in *Proceedings of Australian Telecommunication Networks and Applications Conference (ATNAC)*, 2006. [13](#), [16](#), [53](#)
- [34] Bartosz Jankowski, Wojciech Mazurczyk, and Krzysztof Szczypiorski, “Information hiding using improper frame padding”, in *Telecommunications*

- Network Strategy and Planning Symposium (NETWORKS), 2010 14th International*. 2010, pp. 1–6, IEEE. [13](#), [29](#)
- [35] D. Anthony, D. Johnson, P. Lutz, and B. Yuan, “A behavior based covert channel within anti-virus updates”, 2012. [14](#)
- [36] IPv6 Now Pty Ltd, “Ipv6 rfc and standards working groups”, 2015. [14](#), [69](#)
- [37] Steven James Murdoch author and degree granting institution. University of Cambridge, “Covert channel vulnerabilities in anonymity systems”, 2008., Cataloging Source: Uk eng rda Uk Uk; Bibliography Note: Includes bibliographical references.; Source: EThOS ethos.mds.20140901.mrc. [14](#), [31](#), [37](#), [165](#)
- [38] Lilia Frikha, Zouheir Trabelsi, and Sami Tabbane, “Simulation, optimisation and integration of covert channels, intrusion detection and packet filtering systems”, in *Information Infrastructure Symposium, 2009. GIIS'09. Global*. IEEE, 2009, pp. 1–4. [14](#)
- [39] Pierre Allix, “Covert channels analysis in tcp/ip networks”, *IFIPS School of Engineering, University of Paris-Sud XI, Orsay, France*, 2007. [14](#), [15](#)
- [40] Drago Zagar and Kresimir Grgic, “Ipv6 security threats and possible solutions”, in *Automation Congress, 2006. WAC'06. World*. 2006, pp. 1–7, IEEE. [14](#), [22](#), [28](#)
- [41] Joe Abley, Pekka Savola, and George Neville-Neil, “Deprecation of type 0 routing headers in ipv6”, *draft-ietf-ipv6-deprecate-rh0-01 (work in progress)*, 2007. [14](#), [31](#)
- [42] Steven James Murdoch, *Covert channel vulnerabilities in anonymity systems*, PhD thesis, University of Cambridge, 2008. [14](#), [15](#)
- [43] Sebastian Zander, Grenville Armitage, and Philip Branch, “Covert channels in the ip time to live field”, in *Proceedings of Australian Telecommunication Networks and Applications Conference (ATNAC)*, 2006. [14](#), [165](#)

- [44] Andrei Sabelfeld and Andrew C Myers, “Language-based information-flow security”, *Selected Areas in Communications, IEEE Journal on*, vol. 21, no. 1, pp. 5–19, 2003. [14](#), [15](#)
- [45] Kamran Ahsan and Deepa Kundur, “Practical data hiding in tcp/ip”, in *Proc. Workshop on Multimedia Security at ACM Multimedia*, 2002, vol. 2. [14](#), [16](#)
- [46] Abdulrahman Salih, Xiaoqi Ma, and Evtim Peytchev, “Detection and classification of covert channels in ipv6 using enhanced machine learning”, *Proc. of The International Conference on Computer Technology and Information Systems, ICCTIS, 2015*. [15](#), [17](#), [52](#), [56](#), [57](#), [71](#), [165](#), [168](#)
- [47] Hassan Alsaffar and Daryl Johnson, “Covert channel using the ip timestamp option of an ipv4 packet”, in *The International Conference on Electrical and Bio-medical Engineering, Clean Energy and Green Computing*. The Society of Digital Information and Wireless Communication, 2015, pp. 48–51. [15](#)
- [48] Carlos E. Caicedo, James BD Joshi, and Summit R. Tuladhar, “Ipv6 security challenges”, *Computer*, , no. 2, pp. 36–42, 2009. [15](#)
- [49] Abdur Rahim Choudhary, “In-depth analysis of ipv6 security posture”, in *2009 5th International Conference on Collaborative Computing: Networking, Applications and Worksharing*, 2009. [15](#), [18](#), [23](#), [25](#), [27](#), [29](#), [32](#)
- [50] Fernando Gont, “Security assessment of the ipv6 flow label”, 2012. [15](#), [26](#), [56](#), [71](#), [122](#)
- [51] Christopher Abad, “Ip checksum covert channels and selected hash collision”, *University of California, USA*, <http://downloads.securityfocus.com/library/ipccc.pdf>, 2001. [15](#), [29](#)
- [52] Craig H Rowland, “Covert channels in the tcp/ip protocol suite”, *First Monday*, vol. 2, no. 5, 1997. [15](#), [16](#)
- [53] Serdar Cabuk, Carla E Brodley, and Clay Shields, “Ip covert timing channels: design and detection”, in *Proceedings of the 11th ACM conference on*

- Computer and communications security*. ACM, 2004, pp. 178–187. [15](#), [17](#), [31](#)
- [54] Sergio D. Servetto and Martin Vetterli, “Codes for the fold-sum channel”, in *Conference on Information Sciences and Systems, The Johns Hopkins*, 2001. [15](#)
- [55] Johanna Ullrich, Katharina Krombholz, Heidelinde Hobel, Adrian Dabrowski, and Edgar Weippl, “Ipv6 security: attacks and countermeasures in a nutshell”, *Magdeburger Journal zur Sicherheitsforschung*, vol. 1, pp. 514–529, 2015. [15](#), [17](#), [43](#)
- [56] Alex Dyatlov and Simon Castro, “Exploitation of data streams authorized by a network access control system for arbitrary data transfers: tunneling and covert channels over the http protocol”, *Grayworld, USA*, <http://grayworld.net/projects/papers/html/covert-paper.html>, 2003. [15](#), [17](#)
- [57] Szabolcs Szigeti and Pálter Risztics, “Will ipv6 bring better security?”, in *Euromicro Conference, 2004. Proceedings. 30th.* pp. 532–537, IEEE. [15](#), [28](#)
- [58] S Dutta, PK Mishra, GM Prasad, S Shukla, and SK Chauhya, “Internet protocols: Ipv4 vis a vis ipv6”, *Asian Journal of Information Technology*, vol. 11, no. 3, pp. 100–107, 2012. [16](#), [17](#), [25](#)
- [59] Theodore G Handel and Maxwell T Sandford II, “Hiding data in the osi network model”, in *Information Hiding*. Springer, 1996, pp. 23–38. [16](#)
- [60] J. Ullrich, K. Krombholz, H. Hobel, A. Dabrowski, and E. Weippl, “Ipv6 security: Attacks”, 2015. [16](#), [37](#)
- [61] A. Hintz, “Covert channels in tcp and ip headers”, *Presentation at DEFCON*, vol. 10, 2002. [16](#), [17](#)
- [62] Zouheir Trabelsi and Imad Jawhar, “Covert file transfer protocol based on the ip record route option”. [16](#), [21](#), [23](#)

- [63] Sean Convery and Darrin Miller, “Ipv6 and ipv4 threat comparison and best-practice evaluation (v1. 0)”, *Presentation at the 17th NANOG*, vol. 24, 2004. [16](#), [28](#)
- [64] Joanna Rutkowska, “The implementation of passive covert channels in the linux kernel”, in *Chaos Communication Congress, Chaos Computer Club eV*, 2004. [16](#)
- [65] Tom Dunigan, “Internet steganography”, Tech. Rep., Technical report, Oak Ridge National Laboratory (Contract No. DE-AC05-96OR22464), Oak Ridge, Tennessee, 1998. [16](#)
- [66] Haipeng Qu, Purui Su, and Dengguo Feng, “A typical noisy covert channel in the ip protocol”, in *Security Technology, 2004. 38th Annual 2004 International Carnahan Conference on*. 2004, pp. 189–192, IEEE. [16](#)
- [67] Zbigniew Kwecka, *Application layer covert channel analysis and detection.*, PhD thesis, Edinburgh Napier University, 2006. [17](#)
- [68] Maarten Van Horenbeeck, “Deception on the network: thinking differently about covert channels”, 2006. [17](#)
- [69] Tim Chown, “Use of vlans for ipv4-ipv6 coexistence in enterprise networks”, 2006. [17](#)
- [70] Hayoung Oh, Kijoon Chae, HyoChan Bang, and JungChan Na, “Comparisons analysis of security vulnerabilities for security enforcement in ipv4/ipv6”, in *Advanced Communication Technology, 2006. ICACT 2006. The 8th International Conference*. 2006, vol. 3, pp. 1583–1585, IEEE. [18](#)
- [71] Franjo Majstor, “Does ipv6 protocol solve all security problems of ipv4?”, *Information Security Solutions Europe*, pp. 7–9, 2003. [18](#), [28](#)
- [72] Amirhossein Moravejosharieh, Hero Modares, and Rosli Salleh, “Overview of mobile ipv6 security”, in *Intelligent Systems, Modelling and Simulation (ISMS), 2012 Third International Conference on*. 2012, pp. 584–587, IEEE. [18](#)

- [73] IPv6 Task Force, “Technical and economic assessment of internet protocol version 6 (ipv6)”, *White Paper, US Department of Commerce, National Institute of Standards and Technology, National Telecommunications and Information Administration*, 2006. [20](#), [21](#)
- [74] Steffen Wendzel, Sebastian Zander, Bernhard Fechner, and Christian Herdin, “Pattern-based survey and categorization of network covert channel techniques”, *ACM Computing Surveys (CSUR)*, vol. 47, no. 3, pp. 50, 2015. [21](#), [26](#), [38](#), [39](#), [41](#), [48](#), [52](#), [55](#), [174](#)
- [75] Prabhaker Mateti, “Tcp/ip suite”, *The Internet Encyclopedia*, 2006. [21](#)
- [76] Steve Deering, “Ip version 6 addressing architecture”, Tech. Rep., 2006. [21](#)
- [77] Norka B. Lucena, Grzegorz Lew, and Steve J. Chapin, “Eliminating covert channels in ipv6 with network-aware active wardens”. [22](#), [25](#), [33](#)
- [78] Qinwen Hu and Brian Carpenter, “Survey of proposed use cases for the ipv6 flow label”, 2011. [22](#), [25](#), [26](#), [31](#)
- [79] Dewan Farid, Jerome Darmont, Nouria Harbi, Huu Hoa Nguyen, and Mohammad Zahidur Rahman, “Adaptive network intrusion detection learning: attribute selection and classification”, in *International Conference on computer systems Engineering (ICCSE 2009)*, 2009, p. TH60000. [22](#), [41](#)
- [80] Jonathan Hui and Richard Kelsey, “Multicast protocol for low-power and lossy networks (mpl)”, Tech. Rep., 2016. [23](#)
- [81] Abdur Rahim Choudhary and Alan Sekelsky, “Securing ipv6 network infrastructure: A new security model”, in *Technologies for Homeland Security (HST), 2010 IEEE International Conference on*. IEEE, 2010, pp. 500–506. [23](#), [27](#)
- [82] Suresh Krishnan, “The case against hop-by-hop options”, *Work in Progress*, 2010. [25](#)

- [83] Piet Van Mieghem, Hans De Neve, and Fernando Kuipers, “Hop-by-hop quality of service routing”, *Computer Networks*, vol. 37, no. 3, pp. 407–423, 2001. [25](#)
- [84] Cynthia E. Martin and Jeffrey H. Dunn, “Internet protocol version 6 (ipv6) protocol security assessment”, in *Military Communications Conference, 2007. MILCOM 2007. IEEE.* 2007, pp. 1–7, IEEE. [25](#), [28](#), [30](#), [31](#), [32](#), [34](#)
- [85] Nishant D Rohankar, AV Deorankar, and Dr PN Chatur, “A review of literature on design and detection of network covert channel”, *International Journal of Engineering Science and Innovative Technology (IJESIT) Volume*, vol. 1. [25](#), [28](#)
- [86] D Anthony, D Johnson, P Lutz, and B Yuan, “A behavior based covert channel within anti-virus updates”, 2012. [25](#)
- [87] Geoffrey Ackerman, Daryl Johnson, and Bill Stackpole, “Covert channel using icmpv6 and ipv6 addressing”, in *Proceedings of the International Conference on Security and Management (SAM)*. 2015, p. 63, The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp). [26](#), [27](#), [36](#)
- [88] Joe Abley, Pekka Savola, and George Neville-Neil, “Deprecation of type 0 routing headers in ipv6”, Tech. Rep., 2007. [26](#)
- [89] Kristian Stokes, Bo Yuan, Daryl Johnson, and Peter Lutz, “Icmp covert channel resiliency”, in *Technological Developments in Networking, Education and Automation*, pp. 503–506. Springer, 2010. [27](#), [36](#)
- [90] Russell Bradford, *The Art of Computer Networking*, Pearson Education, 2007. [27](#), [36](#)
- [91] Douglas E Comer, *Computer networks and internets*, Prentice Hall Press, 2008. [27](#)
- [92] RP Murphy, “Ipv6/icmpv6 covert channels”, *Las Vegas: Defcon*, 2006. [27](#), [29](#)

- [93] Alexandru Carp, Andreea Soare, and Razvan Rughini, “Practical analysis of ipv6 security auditing methods”, in *Roedunet International Conference (RoEduNet), 2010 9th.* 2010, pp. 36–41, IEEE. 27
- [94] Susan Thomson, Thomas Narten, and T. Jinmei, ”, *IPv6 Stateless Address Autoconfiguration.IETF*, 2007. 28
- [95] Szabolcs Szigeti et al., “Will ipv6 bring better security?”, in *null.* IEEE, 2004, pp. 532–537. 28, 30, 34
- [96] Pekka Nikander, James Kempf, and Erik Nordmark, “Ipv6 neighbor discovery (nd) trust models and threats”, Tech. Rep. 28
- [97] Mohit Wadhwa and Suresh Kumar, “Security flaws common in ipv4/ipv6 & security issues in ipv6: A study”, 2011. 28, 37
- [98] Dequan Yang; Xu Song; Qiao Guo, “Security on ipv6”, in *Advanced Computer Control (ICACC), 2010 2nd International Conference on*, 2010 2nd International Conference on Advanced Computer Control (ICACC), Ed. mar 2010, vol. 3, pp. 323 – 326, IEEE. 30
- [99] Szabolcs Szigeti, “Will ipv6 bring better security?”, 2004, pp. 532–537, IEEE. 31
- [100] Phrack Magazine, “7 (49), file 06 of 16 [project loki]”. 31
- [101] Phrack Magazine, “Volume seven, issue forty-nine, file 06 of 16,[project loki]”. 31
- [102] IPv6 Task Force, “Technical and economic assessment of internet protocol version 6 (ipv6)”, *White Paper, US Department of Commerce, National Institute of Standards and Technology, National Telecommunications and Information Administration*, 2006. 36
- [103] Scott Hogg and Eric Vyncke, *IPv6 security*, Pearson Education, 2008. 36, 54, 55

- [104] Javed Akhtar Khan and Nitesh Jain, “A survey on intrusion detection systems and classification techniques”, 2016. [38](#)
- [105] J Amudhavel, V Brindha, B Anantharaj, P Karthikeyan, B Bhuvaneshwari, M Vasanthi, D Nivetha, and D Vinodha, “A survey on intrusion detection system: State of the art review”, *Indian Journal of Science and Technology*, vol. 9, no. 11, 2016. [38](#), [39](#)
- [106] Deepika P Vinchurkar and Alpa Reshamwala, “A review of intrusion detection system using neural network and machine learning technique”, *International Journal of Engineering Science and Innovative Technology (IJESIT)*, vol. 1, no. 2, pp. 54–63, 2012. [38](#), [39](#), [40](#), [42](#)
- [107] Mahdi Zamani and Mahnush Movahedi, “Machine learning techniques for intrusion detection”, *arXiv preprint arXiv:1312.2177*, 2013. [38](#)
- [108] Mark Handley, Vern Paxson, and Christian Kreibich, “Network intrusion detection: Evasion, traffic normalization, and end-to-end protocol semantics.”, in *USENIX Security Symposium*, 2001, pp. 115–131. [39](#)
- [109] Steven Gianvecchio and Haining Wang, “Detecting covert timing channels: an entropy-based approach”, in *Proceedings of the 14th ACM conference on Computer and communications security*. ACM, 2007, pp. 307–316. [39](#)
- [110] M. Hall, “Correlation-based feature selection for discrete and numeric class machine learning”, *proceedings of the 7th intentional conference on machine learning, stanford university*, 2000. [39](#), [46](#)
- [111] Rita McCue, “A comparison of the accuracy of support vector machine and naive bayes algorithms in spam classification”, 2009. [39](#), [45](#)
- [112] Yogita B. Bhavsar and Kalyani C. Waghmare, “Intrusion detection system using data mining technique: Support vector machine”, *International Journal of Emerging Technology and Advanced Engineering*, vol. 3, no. 3, pp. 581–586, 2013. [39](#), [44](#), [46](#)

- [113] Sundus Juma, ZAITON MUDA, MA Mohamed, and WARUSIA YASSIN, “Machine learning techniques for intrusion detection system: A review.”, *Journal of Theoretical & Applied Information Technology*, vol. 72, no. 3, 2015. [39](#), [41](#)
- [114] Ozgur Depren, Murat Topallar, Emin Anarim, and M Kemal Ciliz, “An intelligent intrusion detection system (ids) for anomaly and misuse detection in computer networks”, *Expert systems with Applications*, vol. 29, no. 4, pp. 713–722, 2005. [39](#), [41](#), [166](#)
- [115] Cai Zhiyong and Zhang Yong, “Entropy based taxonomy of network convert channels”, in *Power Electronics and Intelligent Transportation System (PEITS), 2009 2nd International Conference on.* 2009, vol. 1, pp. 451–455, IEEE. [39](#)
- [116] James P Anderson et al., “Computer security threat monitoring and surveillance”, Tech. Rep., Technical report, James P. Anderson Company, Fort Washington, Pennsylvania, 1980. [40](#)
- [117] Dewan Md Farid, Nouria Harbi, and Mohammad Zahidur Rahman, “Combining naive bayes and decision tree for adaptive intrusion detection”, *arXiv preprint arXiv:1005.4496*, 2010. [40](#)
- [118] Stephen E Smaha, “Haystack: An intrusion detection system”, in *Aerospace Computer Security Applications Conference, 1988., Fourth.* IEEE, 1988, pp. 37–44. [40](#)
- [119] Debra Anderson, Thane Frivold, and Alfonso Valdes, “Next-generation intrusion detection expert system (nides): A summary”, 1995. [40](#)
- [120] JBD Caberera, B Ravichandran, and Raman K Mehra, “Statistical traffic modeling for network intrusion detection”, in *Modeling, Analysis and Simulation of Computer and Telecommunication Systems, 2000. Proceedings. 8th International Symposium on.* IEEE, 2000, pp. 466–473. [40](#)

- [121] Alfonso Valdes and Keith Skinner, “Adaptive, model-based monitoring for cyber attack detection”, in *International Workshop on Recent Advances in Intrusion Detection*. Springer, 2000, pp. 80–93. [40](#), [41](#)
- [122] Dit-Yan Yeung and Yuxin Ding, “Host-based intrusion detection using dynamic and static behavioral models”, *Pattern recognition*, vol. 36, no. 1, pp. 229–243, 2003. [40](#)
- [123] John E Dickerson and Julie A Dickerson, “Fuzzy network profiling for intrusion detection”, in *Fuzzy Information Processing Society, 2000. NAFIPS. 19th International Conference of the North American*. IEEE, 2000, pp. 301–306. [40](#)
- [124] Wenke Lee and Dong Xiang, “Information-theoretic measures for anomaly detection”, in *Security and Privacy, 2001. S&P 2001. Proceedings. 2001 IEEE Symposium on*. IEEE, 2001, pp. 130–143. [41](#)
- [125] Jake Ryan, Meng-Jang Lin, and Risto Miikkulainen, “Intrusion detection with neural networks”, *Advances in neural information processing systems*, pp. 943–949, 1998. [41](#)
- [126] Taeshik Shon and Jongsub Moon, “A hybrid machine learning approach to network anomaly detection”, *Information Sciences*, vol. 177, no. 18, pp. 3799–3821, 2007. [41](#), [46](#)
- [127] Taeshik Sohn, J Seo, and Jongsub Moon, “A study on the covert channel detection of tcp/ip header using support vector machine”, in *ICICS*. Springer, 2003, pp. 313–324. [41](#)
- [128] George H John and Pat Langley, “Estimating continuous distributions in bayesian classifiers”, in *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 1995, pp. 338–345. [41](#)
- [129] Nahla Ben Amor, Salem Benferhat, and Zied Elouedi, “Naive bayes vs decision trees in intrusion detection systems”, in *Proceedings of the 2004 ACM symposium on Applied computing*. ACM, 2004, pp. 420–424. [41](#), [42](#)

- [130] W Timothy Strayer, David Lapsely, Robert Walsh, and Carl Livadas, “Botnet detection based on network behavior”, in *Botnet Detection*, pp. 1–24. Springer, 2008. [41](#)
- [131] Matija Stevanovic and Jens Myrup Pedersen, “Machine learning for identifying botnet network traffic”, 2013. [41](#)
- [132] Abdulrahman Salih, Xiaoqi Ma, and Evtim Peytchev, “New intelligent heuristic algorithm to mitigate security vulnerabilities in ipv6”, vol. 4, October. [42](#), [52](#), [57](#), [167](#)
- [133] Jonathan Palmer, “Naive bayes classification for intrusion detection using live packet capture”, *Data Mining in Bioinformatics, Spring*, 2011. [42](#), [44](#)
- [134] Nutan Farah Haq, Abdur Rahman Onik, Md Avishek, Khan Hridoy, Musharrat Rafni, Faisal Muhammad Shah, and Dewan Md Farid, “Application of machine learning approaches in intrusion detection system: a survey”, *International Journal of Advanced Research in Artificial Intelligence*, 2015. [42](#), [45](#), [46](#)
- [135] Dewan Md Farid, Nouria Harbi, Suman Ahmmed, Md Zahidur Rahman, and Chowdhury Mofizur Rahman, “Mining network data for intrusion detection through naïve bayesian with clustering”, in *International Conference on Computer, Electrical, System Science, and Engineering (ICCESSE’10)*, Paris, France, 2010, vol. 36. [42](#), [45](#)
- [136] Ron Kohavi, “Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid.”, in *KDD*. Citeseer, 1996, vol. 96, pp. 202–207. [42](#), [45](#), [67](#)
- [137] Mr Manish Jain and Vineet Richariya, “An improved techniques based on naive bayesian for attack detection 1”, 2012. [43](#)
- [138] Dewan Md Farid, Nouria Harbi, Suman Ahmmed, Md Zahidur Rahman, and Chowdhury Mofizur Rahman, “Mining network data for intrusion detection through naïve bayesian with clustering”, in *International Conference on Computer, Electrical, System Science, and Engineering (ICCESSE’10)*, Paris, France, 2010, vol. 36. [43](#), [45](#), [46](#)

- [139] Manish Jain and Vineet Richariya, “An improved techniques based on naive bayesian for attack detection 1”, 2012. [43](#), [45](#)
- [140] Jonathan Palmer, “Naive bayes classification for intrusion detection using live packet capture”, *Data Mining in Bioinformatics, Spring*, 2011. [43](#)
- [141] German Florez, SA Bridges, and Rayford B Vaughn, “An improved algorithm for fuzzy data mining for intrusion detection”, in *Fuzzy Information Processing Society, 2002. Proceedings. NAFIPS. 2002 Annual Meeting of the North American*. IEEE, 2002, pp. 457–462. [44](#), [168](#)
- [142] Ashraf M. Kibriya, Eibe Frank, Bernhard Pfahringer, and Geoffrey Holmes, *Multinomial naive bayes for text categorization revisited*, pp. 488–499, AI 2004: Advances in Artificial Intelligence. Springer, 2004. [44](#)
- [143] Chih-Fong Tsai, Yu-Feng Hsu, Chia-Ying Lin, and Wei-Yang Lin, “Intrusion detection by machine learning: A review”, *Expert Systems with Applications*, vol. 36, no. 10, pp. 11994–12000, 2009. [45](#), [46](#)
- [144] J. Ross Quinlan, “Induction of decision trees”, *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986. [46](#)
- [145] Dewan Md Farid, Nouria Harbi, and Mohammad Zahidur Rahman, “Combining naive bayes and decision tree for adaptive intrusion detection”, *arXiv preprint arXiv:1005.4496*, 2010. [46](#), [53](#), [58](#)
- [146] Melanie Middlemiss and Grant Dick, “Feature selection of intrusion detection data using a hybrid genetic algorithm/knn approach”, in *Design and application of hybrid intelligent systems*. IOS Press, 2003, pp. 519–527. [46](#)
- [147] Yogita B Bhavsar and Kalyani C Waghmare, “Intrusion detection system using data mining technique: Support vector machine”, *International Journal of Emerging Technology and Advanced Engineering*, vol. 3, no. 3, pp. 581–586, 2013. [46](#), [53](#)
- [148] Serdar Cabuk, Carla E. Brodley, and Clay Shields, “Ip covert timing channels: design and detection”, in *Proceedings of the 11th ACM conference on Computer and communications security*. 2004, pp. 178–187, ACM. [49](#)

- [149] Norika B. Lucena, Grzegorz Lewandowski, and Steve J. Chapin, “Covert channels in ipv6”, in *Privacy Enhancing Technologies*. 2006, pp. 147–166, Springer. [49](#), [89](#)
- [150] Gustavus J Simmons, “The prisoners’ problem and the subliminal channel1”, pp. 51–67, 1998. [50](#)
- [151] Johanna Ullrich, Katharina Krombholz, Heidelinde Hobel, Adrian Dabrowski, and Edgar Weippl, “Ipv6 security: attacks and countermeasures in a nutshell”, *Magdeburger Journal zur Sicherheitsforschung*, vol. 1, pp. 514–529, 2015. [53](#)
- [152] Mrutyunjaya Panda, Ajith Abraham, and Manas Ranjan Patra, “Discriminative multinomial naive bayes for network intrusion detection”, in *Information Assurance and Security (IAS), 2010 Sixth International Conference on*. 2010, pp. 5–10, IEEE. [53](#)
- [153] Maher Salem and Ulrich Buehler, “Mining techniques in network security to enhance intrusion detection systems”, *arXiv preprint arXiv:1212.2414*, 2012. [53](#), [54](#), [55](#), [56](#), [58](#), [63](#)
- [154] Steffen Wendzel, “The problem of traffic normalization within a covert channel’s network environment learning phase.”, in *Sicherheit*, 2012, vol. 12, pp. 149–161. [55](#)
- [155] Rusty Russell, “Linux 2.4 packet filtering howto”, <http://netfilter.samba.org/>, 2002. [58](#)
- [156] Danny Roobaert, Grigoris Karakoulas, and Nitesh V Chawla, “Information gain, correlation and support vector machines”, in *Feature extraction*, pp. 463–470. Springer, 2006. [66](#)
- [157] Chotirat Ann Ratanamahatana and Dimitrios Gunopulos, “Scaling up the naive bayesian classifier: Using decision trees for feature selection”, 2002. [67](#)
- [158] Zdravko Markov and Ingrid Russell, “An introduction to the weka data mining system”, in *ACM SIGCSE Bulletin*. ACM, 2006, vol. 38, pp. 367–368. [69](#)

- [159] Peter Backs, Steffen Wendzel, and James Keller, “Dynamic routing in covert channel overlays based on control protocols”, in *Internet Technology And Secured Transactions, 2012 International Conference for.* 2012, pp. 32–39, IEEE. [77](#)
- [160] Philip Miller, *TCP/IP: The Ultimate Protocol Guide*, vol. 2, Universal-Publishers, 2009. [93](#), [134](#)
- [161] Denis Zuev and Andrew W Moore, “Traffic classification using a statistical approach”, in *Passive and Active Network Measurement*, pp. 321–324. Springer, 2005. [95](#)
- [162] Kenneth E Iverson, “A programming language”, in *Proceedings of the May 1-3, 1962, spring joint computer conference.* ACM, 1962, pp. 345–351. [106](#)
- [163] Jonas Taftø Rødfoss, “Comparison of open source network intrusion detection systems”, *Network and System Administration Oslo University College*, 2011. [114](#)
- [164] Jason C Neumann, *The Book of GNS3: Build Virtual Network Labs Using Cisco, Juniper, and More*, No Starch Press, 2015. [118](#), [180](#)
- [165] Firas Najjar and Mohammad M Kadhum, “Reliable behavioral dataset for ipv6 neighbor discovery protocol investigation”, in *IT Convergence and Security (ICITCS), 2015 5th International Conference on.* IEEE, 2015, pp. 1–5. [118](#)
- [166] Nottingham Trent University., “Quality handbook and regulations, section 11”, 2014. [119](#)
- [167] S. Jiang J. Rajahalme S. Amante, B. Carpenter, “Ipv6 flow label specification”, <https://tools.ietf.org/html/rfc6437>, 2011. [127](#)
- [168] A. Ghanwani. N. So B. Khasnabish R. Krishnan, L. Yong, “Mechanisms for optimizing link aggregation group (lag) and equal-cost multipath (ecmp) component link utilization in networks”, <https://tools.ietf.org/html/rfc7424>, 2015. [127](#)

- [169] B. Carpenter S. Deering J. Rajahalme, A. Conta, “Ipv6 flow label specification”, <https://tools.ietf.org/html/rfc3697>, 2004. 128
- [170] Lewis Anderson, Nevil Brownlee, and B Carpenter, “Comparing hash function algorithms for the ipv6 flow label”, *Computer Science Technical Reports*, 2012. 128
- [171] Jeff Schneider and Andrew W Moore, *A locally weighted learning tutorial using vizer 1.0*, Carnegie Mellon University, the Robotics Institute, 2000. 149
- [172] Ron Kohavi et al., “A study of cross-validation and bootstrap for accuracy estimation and model selection”, in *Ijcai*, 1995, vol. 14, pp. 1137–1145. 149
- [173] Dorothy E Denning, “An intrusion-detection model”, *IEEE Transactions on software engineering*, , no. 2, pp. 222–232, 1987. 166
- [174] David Endler, “Intrusion detection. applying machine learning to solaris audit data”, in *Computer Security Applications Conference, 1998. Proceedings. 14th Annual. IEEE*, 1998, pp. 268–279. 166
- [175] Cheng Xiang, Png Chin Yong, and Lim Swee Meng, “Design of multiple-level hybrid classifier for intrusion detection system using bayesian clustering and decision trees”, *Pattern Recognition Letters*, vol. 29, no. 7, pp. 918–924, 2008. 166
- [176] Mansour Esmaili, Bala Balachandran, Reihaneh Safavi-Naini, and Josef Pieprzyk, “Case-based reasoning for intrusion detection”, in *Computer Security Applications Conference, 1996., 12th Annual. IEEE*, 1996, pp. 214–223. 166
- [177] Christopher Kruegel, Fredrik Valeur, and Giovanni Vigna, *Intrusion detection and correlation: challenges and solutions*, vol. 14, Springer Science & Business Media, 2004. 167
- [178] Dieter Gollmann, “Computer security”, *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 2, no. 5, pp. 544–554, 2010. 168

- [179] S Selvakani and RS Rajesh, “Genetic algorithm for framing rules for intrusion detection”, *IJCSNS International Journal of Computer Science and Network Security*, vol. 7, no. 11, pp. 285–290, 2007. [168](#)
- [180] Judea Pearl, *Probabilistic reasoning in intelligent systems: networks of plausible inference*, Morgan Kaufmann, 2014. [170](#)
- [181] Patrick A Gilbert and Prabir Bhattacharya, “An approach towards anomaly based detection and profiling covert tcp/ip channels”, in *Information, Communications and Signal Processing, 2009. ICICS 2009. 7th International Conference on*. IEEE, 2009, pp. 1–5. [174](#)
- [182] Stuart Russel and Peter Norvig, “Artificial intelligence: A modern approach, 2003”, *EUA: Prentice Hall*. [178](#)