IMPROVING THE THREAT DETECTION PERFORMANCE OF A NETWORK INTRUSION DETECTION SYSTEM USING A 3-TIER FRAMEWORK

Samira Dinusha Wickrama Senanayake

(179350H)

Dissertation submitted in partial fulfilment of the requirements for the degree Master of Science in Computer Science specializing in Security Engineering.

Department of Computer Science and Engineering

University of Moratuwa

Sri Lanka

May 2021

DECLARATION OF THE CANDIDATE & SUPERVISOR

I declare that this is my own work, and this dissertation does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to University of Moratuwa the non-exclusive right to reproduce and distribute my thesis/dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

Signature: UOM Verified Signature Date: 24-03-2021

The above candidate has carried out research for the Masters Dissertation under my supervision.

Name of the supervisor: Dr C.D. Gamage

Signature of the supervisor: Date: 24-03-2021

Abstract

Information security is becoming more and more critical for data and information. Network security plays a major role in securing the data and systems from Cyber adversaries. It is crucial to detect the dangers actively and implement defences to protect network infrastructure from Cyber-attackers. In this project, we have introduced a way to optimise the threat detection capabilities using Zeek Network Security Monitor and Weka machine learning application. In fact, we have performed a comprehensive study on the evolution of Intrusion Detection Systems (IDS) using the past literature and identified the factors that contributed to both improved performance and limitations in threat detection. We have designed and developed a Network Security Monitoring (NSM) system prototype using Zeek NSM, Elasticsearch, Filebeat and Kibana Stack(EFK stack) and Weka application.

Moreover, our prototype actively performs network surveillance and alerts the user in an event of intrusion. Finally, we have performed a passive machine learning analysis using Random Forrest, K-Nearest Neighbors and Naïve Bayes classifiers on Denial of Service, Reconnaissance and Worm attacks. We have used a sample set of data from the UNSW-NB15 data set for the machine learning analysis activities.

Installation and configuration of open-source applications are not always straightforward, and they could be swamped with cumbersome processes. We have provided foolproof, stepwise guidance to perform the installation and configure of the Zeek and EFK stack at the end of this thesis.

The authors main objective is to design and develop user-friendly security solutions for threat detection using open-source applications. This project is the initial step to achieve that objective.

Keywords: Network Security, NIDS, Zeek NSM, Weka

DEDICATION

I dedicate this work to all my teachers, lecturers and mentors who taught me not to give up when life is challenging and encourage me to be at my best. May they live a long and healthy life!

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to all those who contributed to carry out this study and make this project a success.

Firstly, I would like to thank my research supervisor Dr C.D. Gamage for giving me valuable advice, guidance, and suggestions to carry out this study.

Secondly, I would like to thank my parents, Ms Vindya Munaweera, Mr Duleep Nakandala, Mr Damsenevi Gurudeniya and my MSc. batchmate Mr Pansilu Pitigalaarachchi who encouraged me and supported me in numerous ways for the whole duration of this project to make this study a reality.

Next, I would like to thank my office colleagues at HSBC Data Processing Lanka (Pvt.) Ltd for all the support and advice given to me to accomplish this project. I would like to mention Mr Kasun Attanayake, Mr Chathura Yatawatte, Mr Lasantha Dasanayaka, Mr Yashodharan Sinnathamby, Mr Hamada Packeer, Mr Richard Baines, Mr Gavin Hawkins, and Mr Colin Fawkes for supporting me in my higher educational activities.

Finally, I would like to thank the Zeek open-source community for being the helping-hand when I needed it the most. I should mention and thank Dr Vern Paxson, Mr Yacin Nadji and Mr Pierre Gaulon for all the advice they provided me through the virtual space.

TABLE OF CONTENTS

DECLARATION OF THE CANDIDATE & SUPERVISOR i
Abstract ii
DEDICATION iii
ACKNOWLEDGEMENTS iv
TABLE OF CONTENTS v
LIST OF FIGURES viii
LIST OF TABLES x
LIST OF ABBREVIATIONS xii
LIST OF APPENDICES xiii
1. INTRODUCTION 1
1.1 Motivation 1
1.2 Security in Computer Networks 2
1.2.1 The Internet 2
1.2.2 The World Wide Web (WWW) 2
1.2.3 Information Security 3
1.2.4 Cybersecurity 3
1.2.5 Detecting Anomalies in Computer Networks 4
1.2.6 Introduction to Intrusion Detection Systems (IDS) 7
1.2.7 Open-source NSM Tools and Their Capabilities 13
1.2.8 Zeek Network Monitoring System 14
1.3 Research Gap 14
1.4 Problem Statement 15
1.5 Aspirations of the Authors 15
1.6 Research Question 16
1.7 Report Organisation 16
2. LITERATURE REVIEW 17
2.1 Intrusion Detection System's Role in Securing the Computer Networks 17
2.1.1 TCP/IP Model 18
2.1.2 Intrusion Detection Systems (IDS) 23
2.1.3 Threat Detection Architectures 28
2.1.4 Threat Detection Metrices for IDSs 31
2.2 Cybersecurity Threats and Threat Detection 36
2.2.1 Anomaly Detection Techniques 39

2.2.2 Machine Learning and Intrusion Detection 45
2.2.3 Feature Selection for Intrusion Detection 53
2.2.4 Significant Challenges in Anomaly Detection 55
2.3 Zeek Network Monitoring System 57
2.3.1 Zeek Architecture 60
2.3.2 Zeek Log Files 61
2.4 Summary 63
3. METHODOLOGY 64
3.1 Overview of Proposed Approach 65
3.1.1 Capture and Analyse Network Traffic in the Test Environment 66
3.1.2 Extract and Pre-process Zeek log Files for Machine Learning 66
3.1.3 Perform Passive Machine Learning Analysis with Weka 66
3.2 Tools and Environments 67
3.2.1 Elasticsearch, Logstash and Kibana (ELK) 67
3.2.2 Weka 68
3.2.3 Network-based Data Sets 69
3.2.4 Test-bed Specifications 76
3.3 The Proposed NIDS Solution 77
3.4 Summary 78
4. IMPLEMENTATION 79
4.1 Capture and Analyse Network Traffic in the Test Environment 80
4.2 Extract and Pre-process Zeek log Files for Machine Learning 84
4.2.1 Data Pre-Processing Procedure 84
4.2.2 Feed Data Files to Weka Application 88
4.3 Perform Passive Machine Learning Analysis with Weka 88
4.4 Training ML Models Using a Sample Set from the UNSW-NB15 Data Set 95
4.4.1 Training the ML Model 101
4.5 Summary 103
5. DISCUSSION AND CONCLUSIONS 104
5.1 Findings 104
5.1.1 Discussion on the Random Forrest Classifier Training 105
5.1.2 Discussion on KNN Classifier Training 106
5.1.3 Discussion on Naïve Bayes Classifier Training 107
5.1.4 Discussion on Attribute Ranking 109
5.2 Challenges and Limitations 110

5.2.1 Installation and Configuration Process 110
5.2.2 EFK? Why Not ELK 111
5.2.3 Storage Issues 111
5.2.4 Integrating Machine Learning to Optimise the Detection Rate 111
5.2.5 Insufficient Data to Train Machine Learning Model – Worm Attacks 112
5.2.6 Log File Analysis 112
5.2.7 Real-Time Threat Detection Using Machine Learning 112
5.3 Conclusion 113
5.4 Future Work 113
REFERENCE LIST 115
APPENDICES 122
Appendix A: Zeek Installation & Configuration 122
Appendix B: Elasticsearch Installation & Configuration 128
Appendix C: Filebeat Installation & Configuration 130
Appendix D: Kibana Installation & Configuration 131
Appendix E: Zeek-EFL Integration 132
Appendix F: Weka Installation 139
Appendix G: Data Pre-processing Process 140
Appendix H: Weka – Classifier Outputs 141
Appendix I: Weka – Ranking the Attributes (Features) 150
Appendix J: Weka – Decision Table Predictive Model Output Information 152

LIST OF FIGURES

Figure 1.1 Intrusion Detection System Example 8
Figure 1.2 Host-based Intrusion Detection System Example 9
Figure 1.3 Network-based Intrusion Detection System Example 10
Figure 1.4 Timeline of Zeek's History [65] 15
Figure 2.1 TCP/IP Reference Model 19
Figure 2.2 TCP Header Format 21
Figure 2.3 IP Header Format 22
Figure 2.4 Firewall Example 23
Figure 2.5 Intrusion Prevention System Example 25
Figure 2.6 Three Major Themes for Improving SOC Operations [47] 26
Figure 2.7 IDS Alert Categorisation 32
Figure 2.8 Percentage Compromised by at Least One Successful Attack, by Year 37
Figure 2.9 Number of Cyber Security Incidents Report to SL CERT from 37
Figure 2.10 Growth of the Types of Cybersecurity Incidents in Sri Lanka 38
Figure 2.11 Signature-based Detection 39
Figure 2.12 Anomaly-based Detection 40
Figure 2.13 Classification of Different Learning 46
Figure 2.14 Common Machine Learning Algorithms 47
Figure 2.15 Integrating Machine Learning to a NIDS 51
Figure 2.16 The Number of Papers Published on ML and Cybersecurity from 2010-2018 52
Figure 2.17 Zeek's Internal Architecture 60
Figure 3.1 Test-bed 64
Figure 3.2 Overview of Proposed Approach 65
Figure 3.3 High-level Architecture to Improve Threat Detection Capabilities 77
Figure 4.1 Implementation Plan 79
Figure 4.2 Online Network Traffic Analysis 81
Figure 4.3 Zeek Offline Network Traffic Log Generation 82
Figure 4.4 Offline Network Traffic Analysis by Replaying .pcap file 83
Figure 4.5 Data Pre-Processing Procedure 84
Figure 4.6 Use Case - Create scan_netflow.pcap file 85
Figure 4.7 Use Case - Generate Conn.log using Zeek utilities 86
Figure 4.8 Malicious.csv file 86
Figure 4.9 Use Case - Generate Conn.log for smallFlows.pcap file 86
Figure 4.10 Normal.csv file 87
Figure 4.11 trainset.arff file format 87
Figure 4.12 testset.arff file format 87
Figure 4.13 Feed .arff files to Weka Application (Training Data and Testing Data) 88
Figure 4.14 Weka Explorer GUI 89
Figure 4.15 Results of the Decision Table Algorithm 89
Figure 4.16 Results of the J48 Algorithm 90
Figure 4.17 Decision tree Reliance on the Time Feature 91
Figure 4.18 Results of the J48 Algorithm After the Re-configuration 91
Figure 4.19 Unprocessed testset.arff 92
Figure 4.20 ML Model Predictions 93
Figure 4.21 First 10 instances of test.csv file and the output of predictive model 93

Figure 4.22 Visualising the Predicted Labels for the Test Data Set 94
Figure 4.23 Procedure to Test the Selected Sample Data 98
Figure 4.24 UNSW-NB15 Sample Data Set Testing Framework 99
Figure 4.25 Upload the Master.csv file to Weka 100
Figure 4.26 Relationship Between the Attributes (Features) and the Classes 100
Figure 4.27 Result of the ML Analysis – Random Forrest Classifier 101
Figure 4.28 Result of the ML Analysis – KNN Classifier 102
Figure 4.29 Result of the ML Analysis – Naïve Bayes Classifier 103
Figure 5.1 Weka Interface - Current Relation & Selected Attribute 104
Figure 5.2 Classifier Performance Comparison 108
Figure 7.1 VirtualBox Host-Only Ethernet Adapter Setup 122
Figure 7.2 Setup the Sniffing Interface 123
Figure 7.3 Kibana Configuration - Kibana.yml 132
Figure 7.4 Elasticsearch Configuration - Elasticsearch.yml 133
Figure 7.5 Filebeat Configuration ii - Filebeat.yml 134
Figure 7.6 Filebeat Configuration i - Filebeat.yml 134
Figure 7.7 Filebeat Configuration iii - Filebeat.yml 135
Figure 7.8 zeek.yml File Configuration 136
Figure 7.9 Checking the application status 137
Figure 7.10 Accessing Kibana 138
Figure 7.11 Weka GUI 139

LIST OF TABLES

Table 1.1 Most Common Network Attacks 6
Table 1.2 Comparative Study of Well-known Open-source IDSs 13
Table 2.1 TCP/IP Layers [23] 19
Table 2.2 Layer-specific Protocols in TCP/IP Reference Models [40] 20
Table 2.3 Difference Between IDS and Firewalls 24
Table 2.4 IDS vs. IPS in Nutshell 25
Table 2.5 Categorisation of Intrusion Detection Systems 27
Table 2.6 Comparison of HIDS and NIDS Technologies 29
Table 2.7 Advantageous and Limitations of the Statistical Based Approach 41
Table 2.8 Advantageous and Limitations of the Clustering Approach 41
Table 2.9 Advantageous and Limitations of the Finite State Machine Approach 42
Table 2.10 Advantageous and Limitations of the Classification Based Approach 42
Table 2.11 Advantageous and Limitations of the Information Theoretic-based Approach 43
Table 2.12 Advantageous and Limitations of the Evolutionary Computation Approach 43
Table 2.13 Advantageous and Limitations of the Hybrid Anomaly Detection Approach 44
Table 2.14 Comparison of Intrusion Detection Techniques 44
Table 2.15 Characteristics of SIDS and AIDS 44
Table 2.16 Pros and Cons of ANN 47
Table 2.17 Pros and Cons of SVM 48
Table 2.18 Pros and Cons of KNN 49
Table 2.19 Pros and Cons of Naive Bayes 49
Table 2.20 Pros and Cons of LR 49
Table 2.21 Pros and Cons of Decision Tree 50
Table 2.22 Pros and Cons of K-means 50
Table 2.23 Malware Behavioural Features Collated by Akinrolabu [44] 54
Table 2.24 Identified Features by the Interviews (Akinrolabu) [44] 55
Table 2.25 Logs Related to Network Protocols 62
Table 2.26 Logs Related to Threat Detection 62
Table 2.27 Logs Related to Network Observation 63
Table 3.1 Properties of UNSW-NB15 Data Set 71
Table 3.2 UNSW-NB15 Record Distribution 72
Table 3.3 Features of UNSW-NB15 74
Table 3.4 Test-bed Virtual Machines 76
Table 3.5 Applications in Use 77
Table 4.1 Use Case - Online Network Traffic Analysis 80
Table 4.2 Use Case - Offline Network Traffic Analysis 82
Table 4.3 Use Case - Offline Network Traffic Analysis by Replaying .pcap file 82
Table 4.4 Use Case - Creating .arff Files Using Conn.logs 85
Table 4.5 Confusion Matrix - Decision Table Algorithm 90
Table 4.6 Confusion Matrix - J48 algorithm 91
Table 4.7 Confusion Matrix - J48 algorithm After Re-configuration 92
Table 4.8 Confusion Matrix - Decision Table Predictive Model 94
Table 4.9 UNSW-NB15 No. Records on the Selected Attacks 96
Table 4.10 UNSW-NB15 Features Proposed by ARM Algorithm 97
Table 5.1 Detailed Accuracy by Class - Random Forrest 105

Table 5.2 Confusion Matrix - Random Forrest 105
Table 5.3 Detailed Accuracy by Class - KNN 106
Table 5.4 Confusion Matrix - KNN 106
Table 5.5 Detailed Accuracy by Class - Naïve Bayes 107
Table 5.6 Confusion Matrix - Naïve Bayes 107
Table 5.7 Ranked Attributes Using Weka 109

LIST OF ABBREVIATIONS

DoS	Denial-of-service
EFK	Elasticsearch, Filebeat, Kibana
ELK	Elasticsearch, Logstash, Kibana
HIDS	Host-based Intrusion Detection System
IDS	Intrusion Detection System
IP	Internet Protocol
KNN	K-Nearest Neighbor Algorithm
ML	Machine Learning
NIDS	Network-based Intrusion Detection System
NSM	Network Security Monitor
TCP	Transmission Control Protocol
TSV	Tab-separated Values
WEKA	Waikato Environment for Knowledge Analysis
.arff	Attribute-Relation File Format
.csv	Comma-separated values Format
.pcap	Packet Capture Format

LIST OF APPENDICES

Appendix – A	Zeek Installation & Configuration	122
Appendix – B	Elasticsearch Installation & Configuration	128
Appendix – C	Filebeat Installation & Configuration	130
Appendix – D	Kibana Installation & Configuration	131
Appendix – E	Zeek-EFL Integration	132
Appendix – F	Weka Installation	139
Appendix – G	Data Pre-processing Process	140
Appendix – H	Weka – Classifier Outputs	141
Appendix – I	Weka – Ranking the Attributes (Features)	150
Appendix – J	Weka – Decision Table Predictive Model Output	152

1. INTRODUCTION

1.1 Motivation

The introduction of the personal computer and consequent technologies in the 1970s lead mankind to a new era which is known as the information age or the digital age. The underlying principle of the information age is to provide the ability and the technology to process and transfer information freely and rapidly through the digital space. Information and Communication Technology (ICT) has changed the way we behave in our daily life. Paper-based newspapers, magazines, books, and tabloids have now moved into online newspapers and e-books. Traditional communication methods such as letters, notes, have now evolved to electronic mail, online chats, call conference and video conference to make our daily tasks more convenient. These enhancements in ICT have paved the way for new technologies such as electronic banking facilities, electronic channelling services, cloud computing, Internet of Things (IoT), Artificial Intelligence (AI) which are now indispensable in our daily lives.

Security is always a high priority issue for governments, organisations, and the general public. Dozens of anti-virus applications, encryption methods, firewalls, Intrusion Detection Systems (IDS), Intrusion Prevention Systems (IPS), information security protocols and standards have been introduced and implemented to improve the software security, network security, database security and computer infrastructure security against adversaries and threats. The innovative criminal activities have proven that current security parameters are not sufficient and require more development to defend novel threats.

This thesis is focused on the network security subject area and expected to explore an effective defence mechanism for intrusion detection systems against selected network related attacks.

1.2 Security in Computer Networks

The internet is the world's most extensive computer network. The relationship between the internet and the World Wide Web (WWW) is often misunderstood. Although these words are sometimes used interchangeably, they refer to two distinct technologies.

1.2.1 The Internet

In simple terms, the internet is a network of computer networks. Each network uses a standard set of rules called ‘’network protocols’’ to communicate with each other. Moreover, the internet has masses of connections which are related to the public sector, academic sector, commercial sector. A wide range of communication medium used to communicate over the internet, such as Coaxial cables, Twisted-Pair Cables, Wireless and Fiber-Optic cables. The internet mainly uses Transmission Control Protocol/Internet Protocol (TCP/IP) to communicate with users.

1.2.2 The World Wide Web (WWW)

WWW is also known as the “web” was invented in 1989 by English computer scientist Tim Berners-Lee. The web is a mixture of document, applications, resources, and users that use Hypertext Transfer Protocol (HTTP). In simple terms, WWW, are the web pages users see when they browse the internet using a computer device.

Both the internet and the web require protection against computer criminals and adversarial activities. Two most common terms used in this domain are Information Security and Cybersecurity. These two terms are two different terminologies, let us investigate their descriptions next.

1.2.3 Information Security

Information security or InfoSec is focusing on the procedures and systems used to defend information from unauthorised access, use, disclosure, disruption, modification, perusal, inspection, recording or destruction. For example: If you print out a confidential document and leave it on a public area, it would be an information security breach but not a Cybersecurity breach.

1.2.4 Cybersecurity

Cybersecurity refers to protection against the attacks coming via a network such internet. For example: If the organisation's web-facing server flood with data packets as a result of a Distributed Denial of Service Attack (DDoS), that would constitute a Cyber-attack but not an information security breach.

As the focus of this thesis is Network Security, going forward authors will use Cybersecurity in the following sections of this report.

Securing the network infrastructures has become a crucial aspect as attacks or anomalies in the network can significantly affect the national security and the economy. As a result, detecting network anomalies and attacks until they cause harm to network infrastructures is critical. Before we go any further, it is important to know what the word "anomalies" means. Anomalies, according to Chandola et al., are data patterns that do not adhere to a well-defined notion of normal behaviour [1]. Anomalies, according to Lakhina et al., are abnormal and vital shifts in a network's traffic levels, which can often issue [2]. According to Hoque et al., anomalies are "non-conforming" interesting patterns compared to the well-defined notion of normal behaviour" [3]. These definitions demonstrate that understanding the concept of normality is important for designing anomaly detection solutions.

1.2.5 Detecting Anomalies in Computer Networks

Distinguishing normal and abnormal behaviour out of undefined network data streams is a challenging task in network security. Developing a better anomaly detection system is based on factors such as rapid identification, lower complexity, and the incident handling procedures. Furthermore, accuracy and speed, as well as the timely detection of irregular events, are critical for minimising major service outages, financial losses, and reputational harm [4]. Study of anomaly detection goes back to the early 19th century. Researchers have proposed various methods to detect anomalies. Moreover, statistical models, clustering techniques, machine learning, graph-based methods and many more approaches have developed and introduced to anomaly detection domain via hundreds and thousands of research and development work. A fundamental aspect of classifying anomalies is that they may be malicious or non-malicious, depending on their source. It is important to note that anomalies are not necessarily linked to attacks aimed at causing damage to computer systems or steal information. Anomalies can also generate via human/hardware failure, bugs or private users when demanding heavy traffic usage. Hence anomaly detection is a challenging task. We are expecting to discuss further on the challenges for anomaly detection in Chapter 2.

A series of malicious acts aimed at disrupting, denying, degrading, or destroying information and resources from computer network systems, compromising their confidentiality, integrity, or availability are known as network attacks or network manipulation anomalies. [4].

The confidentiality, integrity or availability or the CIA triad is a model designed to steer policies for Cybersecurity. Let us have a quick look at the theoretical definition of each component.

Confidentiality: Confidentiality protects the individual and guarantees the confidentiality of sensitive data by employing a range of measures to avoid unauthorised disclosure [5].

Integrity: Maintaining integrity assures that precise and complete data is used by the system [6]. Data must not be changed when in transit, and precautions must be taken to ensure that data cannot be altered by unauthorised persons. [7].

Availability: Ensuring timely and reliable access to and use of information [7]. Availability is the protection of the information system from break-down [8].

The network attack arsenal of Cybercriminals is complex and varied. It ranges from basic spam attacks to sensitive network infrastructure intrusion attacks. Popular computer attacks include worms, malicious resource misuse, bug exploits, and unauthorised alteration [4]. Cyber-attack domain is multi-dimensional. It entailed with technical approaches as well as psychological approaches. In fact, attacks like Social Engineering does not require technical skills.

Social engineering is a type of Cyber-attack in which an intruder uses aggressive manipulation or other interpersonal techniques to exploit people in order to gain confidential information [9].

Focussing on the other approaches, which attackers exercise to endanger the functionality of computer systems includes, masquerading attack, exploiting vulnerabilities, abuse of functionality [10]. In Masquerading attacks, the attacker impersonates as authorised users and disrupt the internal network activities [11].

In Abuse of Functionality, the attackers conduct normal activities with the abusive intent, which ultimately cause systems to fail. Example: Denial of Service Attack (DoS)

Network attacks or network anomalies can be classified into different categories based on their characterisation. Fernandes Jr. et al. present a summarised depiction of network attacks variants that a network anomaly detector should identify [4]. Refer Table 1 for more details.

Table 1: Most Common Network Attacks

Network attack categories	Attack	Definition
Social Engineering	Trojan	• A piece of program masquerading as a benign application but comprise with malicious activities. • They do not replicate as viruses and worms do but can be just as destructive
Exploiting Vulnerabilities	Buffer overflow	• Takes over programs through buffer vulnerabilities to execute arbitrary code in order to store more data in a buffer than the buffer can hold. • Can corrupt or overwrite valid data held in a buffer
	Password attack	• Attempts to gain passwords
	Cyber reconnaissance	• Gathers information on network systems and services
	Probe	• It is accomplished before an attacker launches an attack on a given target. • Scans or probes targets the network or host by searching for vulnerabilities, open ports, valid IP addresses, services offered, operating system used, etc.
Masquerading	User to Root (U2R)	• Consists of unauthorised access to local superuser privileges by starting as a regular unprivileged user. • U2R attacks may end in substantial loss of time and money
	Remote-to-Local	• Unauthorised access via a remote machine • Remote to local attacks can be detect using supervised neural network methods
Abuse of Functionalities	Denial of service (DoS)	• Malicious attempts to deny access to shared network resources or services
	Distributed DoS (DDoS)	• DDoS are DoS attacks; they are easy to launch and difficult to locate their source since they are implemented by a group of computers (botnet)
	Distributed reflective DoS (DRDoS)	• Attacks that just cannot be addressed by traditional on-premises solutions. These use legitimate hosts (reflectors) to flood many response packets to the target system by using spoofed IP addresses
	Physical attack	• An endeavour to harm physical components of a computer or network
Other	Virus	• Piece of code inserted into a file or program which replicates itself without the user's permission. • Harmful activities: theft of private information, data corruption, spam messages • A computer virus is a computer program which can propagate itself without the user's consent. Usually, it has malicious intent, e.g., to damage a computer system or acquire information without owners' permission. An antivirus software is used to identify and remove computer viruses [12].
	Worm	• Self-replicating software designed to spread through the network
	Stealthy attack	• Quietly introduced and remain undetected by hiding the evidence of the attacker's actions

The behaviour of the anomaly detection system is depending on the underlying network data source. Anomaly detector developers should select the data source by considering the focus area of the detection system. This helps the anomaly detection system to perform up to the expected level by correctly identifying anomalies. TCP dump, SNMP, IP flow can be listed as popular data source used for anomaly detection.

An intrusion or incident is an event on a host or network that breaches security policies or poses a significant risk of bringing a device into an unauthorised state. Intrusion detection is the method of tracking and analysing device activities in order to detect and report those intrusions. An intrusion detection system or IDS is an automated solution that perform events monitoring, data log analysis, network traffic analysis, and to notify respective personnel/systems in an event of security breach [13].

Information systems are networks and exposed to external attackers through the Internet. Attacks are continuously evolving and becoming sophisticated making them harder to detect and defend. An Intrusion Detection System (IDS) is a major component of the defence mechanism of an information system. An IDS is generally installed and configured through a cumbersome manual process. Next section will provide a high-level introduction on Intrusion Detection Systems and its variants.

1.2.6 Introduction to Intrusion Detection Systems (IDS)

In a nutshell, intrusion detection systems (IDSs) detect attacks on a specific collection of computer properties, ranging from a single desktop PC to a large corporate enterprise network. An American information security researcher Dorothy Elizabeth Denning is one of the pioneer researchers who lay the foundation to current intrusion detection technologies. In 1987 Denning introduced a model for a real-time intrusion detection system which, capable of detecting intrusions and other forms of computer abuse. This model provides profiles for describing typical user activity in terms of metrics and statistical models for the resources handled by the system-files, commands, and devices, as well as rules for gaining information about behaviour from audit reports and detecting behavioural irregularities [14]. Between the late 1980s and

early 1990s, researchers have developed successful detection models using statistical approaches. Between the mid-1990s and the late 1990s, the method of gathering information about normal and irregular behaviour was transformed from a manual to an automated system. Artificial intelligence (AI)-related technologies are currently used for IDS technologies.

Firewalls are basic perimeter defences that are intended to block the majority of malicious traffic. Intrusion detection systems and associated protections serve as eyes to see what makes it through [13]. Figure 1 depicts a high-level diagram of an IDS.

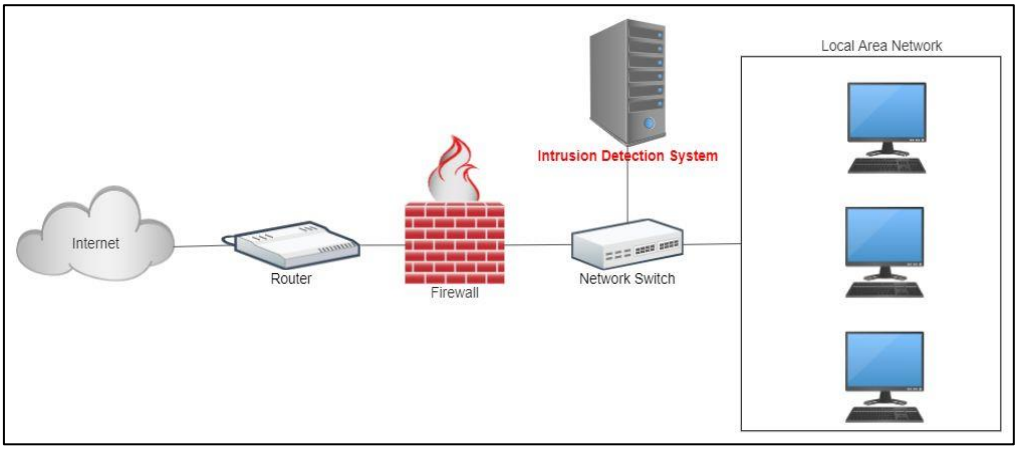


Figure 1: Intrusion Detection System Example

Intrusion Prevention Systems (IPS) vs. Intrusion Detection Systems (IDS)

The role of IPS is to respond in real-time if a threat is detected, the IPS takes actions by modifying its environment; it could modify the firewall by closing a suspicious connection, or reconfigure the router, etc. [15].

On the other hand, IDS is responsible for conducting passive monitoring (In passive monitoring, the IDS detect the anomaly and inform the users), packet capturing and analysing the network traffic. IDS can be configured to inform the servers and the users in an event of possible intrusion.

IDS Types

Host-based Intrusion Detection System (HIDS)

Insider attacks that do not involve network traffic are detected by HIDS [16]. The Hosts-based IDS (HIDS) detects the current intrusion by aggregating and analysing data (system logs or operating system audit trails) from a device that hosts a service, such as a web server; it then engages a local agent who is responsible for monitoring and reporting. If the agent detects any changes in the system's actions or program operation, it will warn the user that there has been an attack [17]. You can get a clear picture on the high-level layout of an HIDS from Figure 2.

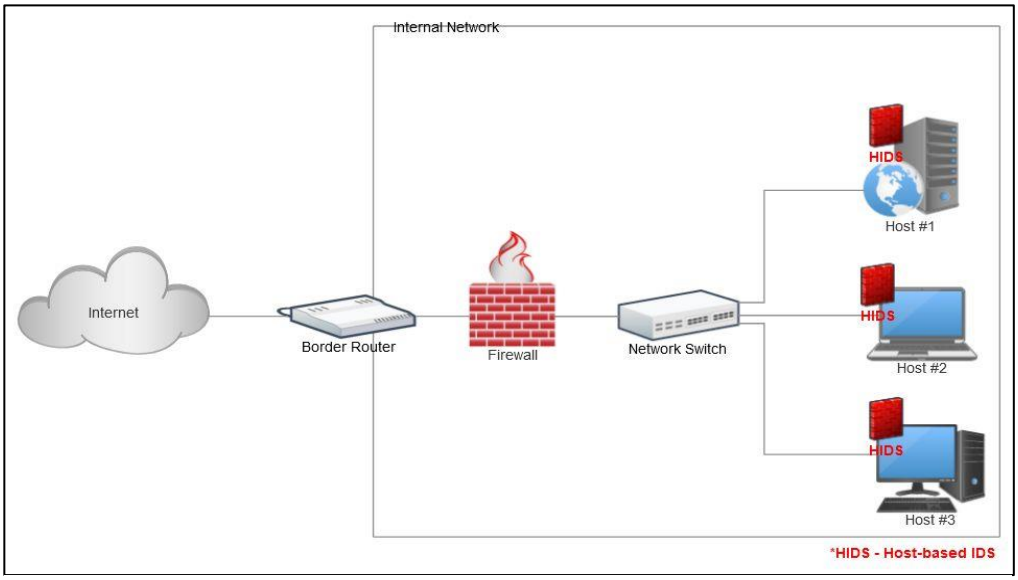


Figure 2: Host-based Intrusion Detection System Example

Network-based Intrusion Detection System (NIDS)

A network intrusion detection system (NIDS) that detects and protects all network nodes by detecting intrusions in network data over network connections. They are made up of a network of sensors that track traffic at different network points. Each sensor conducts a local inspection and alerts a central management console to any

suspicious behaviour. An IDS that is network-based will collect and analyse entire transmitted packets, including payloads, IP addresses, and ports [4]. The high-level architecture of a NIDS is depicted in Figure 3.

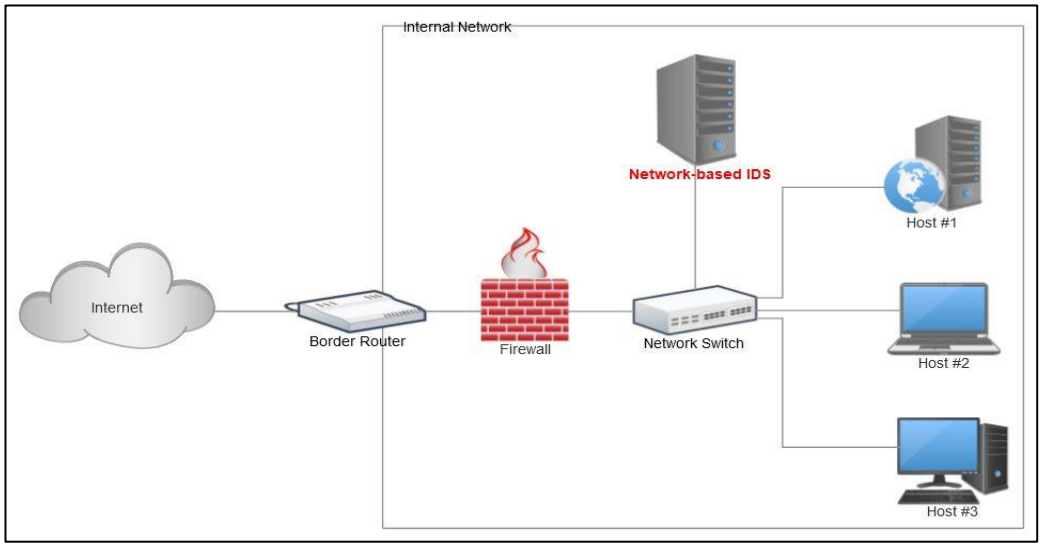


Figure 3: Network-based Intrusion Detection System Example

Hybrid IDS

Hybrid IDSs are created by integrating the functionalities of both network and host-based IDSs and using data from host events and network segments [18]. Hybrid IDSs, on the other hand, may not always indicate better systems. Since various IDS technologies monitor traffic and search for intrusive behaviour in different ways, incorporating and coexisting these technologies in a single system effectively and efficiently is a difficult job [4].

Threat Detection Groups

Signature-based (Misuse Detection)

In anti-virus software and intrusion detection systems, signature-based detection is the most commonly used method. Malware detection has largely focused on conducting static investigations to review the code-structure mark of infections, rather than using element behavioural approaches [19].
A predefined list of documented assaults is used to find interruptions in the signature-based scheme. Even though this arrangement can detect malware in a mobile application, it necessitates daily updates to the predefined signature database. Furthermore, because of the fast-changing nature of portable malware, the signature-based approach is less successful in detecting Zero-day attacks. [20] [21].

Anomaly-based (Anomaly Detection)

Anomaly detection techniques look at how people perceive normal activity in data and mark any exceptions as anomalies. This method of detection causes previously undetected attacks to be detected. These methods, however, are known to generate many false alarms since they flag any deviation as an attack [22].

Specification-based

Stateful protocol analysis is another name for this procedure. The term "stateful" refers to an IDS' ability to comprehend and monitor the state of network, transport, and application protocols that have a state definition [23].
Anomaly detection, unlike signature-based intrusion detection, is capable of detecting novel attacks. However, in practice, anomaly detection is plagued by a high rate of false alarms. Specification-based techniques have been shown to have a low rate of

false alarms, but they are not as successful as anomaly detection in detecting novel attacks, especially network probing and denial-of-service attacks [24].

Hybrid Techniques

Compound detection, also known as hybrid IDSs, uses a mixture of misuse, anomaly, and specification detection techniques [4]. To detect unknown anomalies within network traffic, de Assis et al. proposed a hybrid detection approach that distinguishes a network profile called digital signature of network segment using flow analysis (DSNSF). The discovered anomalous behaviour was then labeled as a DoS, DDoS, flash crowd, or port scan attack using pre-loaded signatures [25].

Challenges in Detecting Anomalies

Intrusion detection techniques are continuously improving, with the aim of improving network and computer infrastructure security and safety. But it does not mean the current anomaly detection systems are effective in detecting threats accurately [26].

One of the main challenges in threat detection domain is due to the high false-positive rates the detection accuracy has to deal with inefficiencies [27]. Moreover, the academic work conducted by Kruegel et al. point out that a higher number of false positives degrade the IDS performances [28]. IDSs require complex network detection configurations to mitigate both false-positives and false-negative detections [29].

Another concern is the low throughput and high cost of existing wideband transmission technologies, which is largely due to the high data rates (Gbps) that characterise them [30]. Furthermore, there is a lack of adequate metrics, evaluation methodologies, and a broad structure for assessing and comparing various IDS techniques. [31] [32]. We will discuss IDSs, related technologies and threat detection techniques in detail in Chapter 2.

1.2.7 Open-source NSM Tools and Their Capabilities

We are using the Zeek application as our NSM system. Before going forward it is important to stress the rationale behind the selection of Zeek as our NSM solutions. Thus, this section covers the motivation behind selecting Zeek for this experiment. Zeek is not the only open-source NSM solution available to use and there are several alternative applications one could use to build his or her NSM solution. In the requirement gathering and feasibility study phase, we have identified unique features related to network security in Zeek, which made us select Zeek for our project work. The following table will present a feature comparison between Zeek and well-known NSM alternatives that are available in the open-source domain.

A comparative study of well-known open-source IDS solutions are given in Table 2 [33] [34] [35]. The following information was taken from related literature.

Table 2: Comparative Study of Well-known Open-source IDSs

Functionality	**Suricata**	**Snort**	**Zeek**
Ram usage	High	Medium	Low
Dropped packets	Medium	High	Low
Offline packet analysis	Yes	Yes	Yes
Support to high-speed networks	Medium	High	High
Develop and modify threat detection scripts/modules (Higher customization)	No	No	Yes
Signature-based detection	Yes	Yes	Yes
Anomaly-based detection	No	No	Yes
Detecting bad traffic	Medium	Medium	High
Payload mutation technique detection	Low	High	High
Logfile readability	Low	Low	High
Good Community Support	Yes	Yes	Yes

As per the positive aspects depicted in Table 2, we believed that Zeek is suitable for our journey in threat detection and threat hunting. Thus, we have selected the Zeek application for our project work.

1.2.8 Zeek Network Monitoring System

Zeek, formerly known as Bro, is an open-source passive network traffic analyser. This program is primarily concerned with network traffic control and the detection of suspicious network activity. Also, outside of the security domain, Zeek validates a wide variety of traffic analysis functions, such as performance assessments and troubleshooting [36]. Zeek was developed by American computer scientist Vern Edward Paxson in 1994.

Zeek Network Security Monitor is a network-dependent analytics framework that transforms network traffic into events that can be used to trigger scripts. This program is similar to an intrusion detection system (IDS). Zeek can use both signature-based and anomaly-based monitoring. Features of Zeek expand from performing forensics activities to actively monitor networks. Zeek application has contributed to filling the gap between academia and operations for over 20 years [37]. The evolution of Zeek IDS shown in Figure 4.

We will be using Zeek application as our network monitoring system (NSM). The underlying technology of this NSM application will be further discussed in Chapter 2.

1.3 Research Gap

The IDSs continuously monitor activities in information systems that consist of software applications, computer systems and network infrastructure. Anomaly-based IDSs can flag as suspicious activities that exceed an allowed threshold or matches a profile of malicious activity. As this detection is probabilistic, IDSs tend to have significant false positives (normal activity flagged as attacks) and false negatives (attacks passed over as normal activity). This results in poor performance of the IDSs thus severely threatening the security of the information systems.

1.4 Problem Statement

The literature on existing intrusion detection system architectures and techniques shows a significant gap between the theoretically achievable and practically achieved performances. This proposed research aims to reduce this gap.

1.5 Aspirations of the Authors

- To examine the existing IDSs and their threat detection technologies.
- To determine the aspects of NIDS design and operation that contribute to an increase in its threat detection capabilities.
- To investigate the possibility of improving the performance of threat detection modules of a NIDS by improving Zeek anomaly detection modules.
- To develop a prototype system to study the practical performance of the threat detection modules of a NIDS.

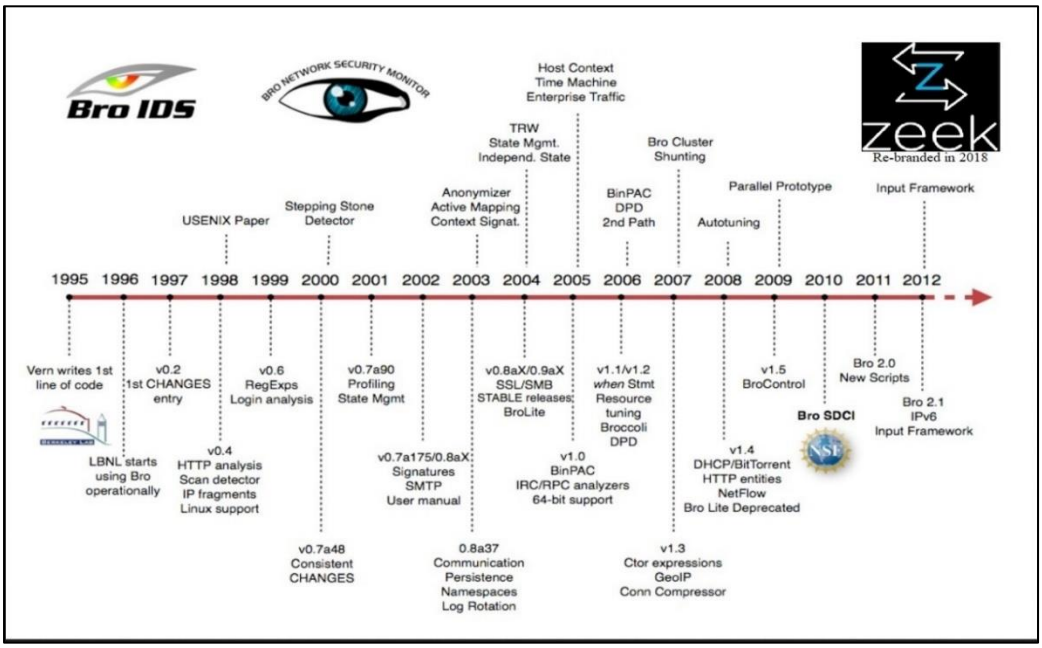


Figure 4: Timeline of Zeek's History *[65]*

1.6 Research Question

Is it possible to design and develop a conceptual framework for exiting Network-based Intrusion Detection System technologies with the intent of improving the performances by incorporating technology that allows continuous learning and detects network-based threats?

1.7 Report Organisation

Chapter 1 – 1. INTRODUCTION: Briefly introduces the IDSs and related technologies, why it is required, and the challenges associated with it along with how this thesis aims to solve these challenges.

Chapter 2 - 2. LITERATURE REVIEW: Study existing IDS technologies and identify the features that contribute to both enhancements and limitations of the performance in IDSs.

Chapter 3 – 3. METHODOLOGY: Improve the threat detection capabilities of the Zeek network monitoring system and enhance threat detection performance by decision table classifier using Weka.

Chapter 4 – 4. IMPLEMENTATION: Build and test a prototype implementation of and NSM incorporating the proposed improvements.

Chapter 5 – 5. DISCUSSION AND CONCLUSIONS: Analyse the findings and discuss the challenges, limitations, and author's future research suggestions.

2. LITERATURE REVIEW

2.1 Intrusion Detection System's Role in Securing the Computer Networks

Network security refers to the defence of a network against Cyber-threats that could jeopardise network availability, misuse network resources, or allow unauthorised access. Moreover, Amrollahi et al. state that network security is a layered security model, and it should consist with a combination of multiple different layers with diverse controls, policies and security devices of Cybersecurity that brought together to strengthen the organisation's network infrastructure. Amrollahi et al. also mention four components, which is important for network security. Namely, physical security, parameter protection, monitoring, and user-awareness [38].

Physical Security

Protecting the IT components from physical damages is essential. Taking necessary actions for protecting IT assets against natural disasters, terrorist attacks, theft and illegal access have become a mandatory task in physical security. Hence, physical security is a fundamentals factor for computer network security. Business Continuity Plans (BCP) and Disaster Recovery (DR) processes prepared as a part of the countermeasures against physical security threats.

Parameter Protection

The parameter defence proposes strategies for separating computer systems from physical devices and the rest of the world. The protection provided by the firewalls, routers and other network security devices protect the network infrastructure by monitoring and blocking the external network traffic.

Monitoring

Monitoring is the first step in detecting and distinguishing between irregular and normal network traffic. It is important to detect the abnormal behaviours sooner and execute countermeasures to minimise the damage.

User Awareness

Humans are the weakest link in the Cybersecurity chain. It is vital to strengthen the knowledge base of the users in terms of Cybersecurity and information security. Proper Cyber-awareness is essential for network security. This can be achieved by providing training and education to the users.

Protocols are used to communicate between networks. The internet protocol suite, or TCP/IP, is the most commonly used model on the internet and other computer networks [23]. We believe that it would be beneficial for our readers to have a brief understanding of the TCP/IP model before going forward in the Intrusion Detection subject area.

2.1.1 TCP/IP Model

TCP/IP is a conceptual model and set of communication protocols. TCP/IP refers to the Transmission Control Protocol and Internet Protocol. [39]. TCP/IP is a network communication protocol that is commonly used around the world.

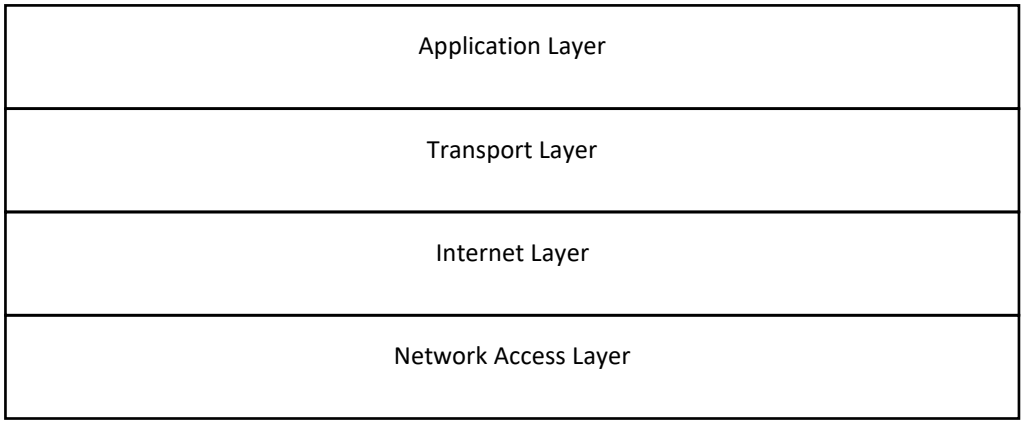


Figure 5: TCP/IP Reference Model

TCP/IP model is entailed with four layers (Figure 5). Namely, application layer, transport layer, internet layer and network access layer. When a sender transmits data over the computer network, data first go through the highest layer (Application layer) and pass through the intermediate layers and reached to the lowest layer (Network access layer) before reaching the receiver's end. In this process, each layer adds information, which later contributes to a successful data transmission. Data receivers end collate incoming data from the Network access layer and pass up towards Application layer. The tasks performed by the four layers from highest to lowest are described in in the Table 3.

Table 3: TCP/IP Layers *[23]*

TCP/IP Model	Protocols
Application layer	This layer sends and receives data form domain name system (DNS), hypertext transfer protocol (HTTP), and simple mail transfer protocol (SMTP) applications. There are a plethora of application layer protocols in use today. Application data is generated and then transferred to the transport layer for further processing, regardless of the protocol in use.
Transport layer	The transport layer enables communication between entities on the source and destination machines. This layer transports application layer services between networks using connection-oriented or connectionless services. The transport layer will optionally ensure communication continuity. Transport layer protocols such as Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) are widely used.
Internet layer	This layer's main goal is to enable a host to insert packets into any network and then ensure that they move to their destination

	independently. Furthermore, this layer is responsible for packet routing across networks.
Network-Access layer	The physical network components' communications are handled by this layer. Ethernet is the most well-known data link layer protocol.

A variety of network protocols have been introduced to facilitate communication between two ends based on the core TCP/IP model. Below the Table 4 lists few protocols in TCP/IP reference model.

Table 4: Layer-specific Protocols in TCP/IP Reference Models *[40]*

TCP/IP Model	Protocols
Application layer	BitTorrent, BootP, DNS, DHCP, FTP, HTTP, HTTPS, NTP, SSH, SMTP, SNMP, Telnet, TFTP, XMPP, SIP, RDP, POP3, IMAP
Transport layer	SCTP, TCP, UDP, SSL
Internet layer	ICMP, IGMP, IP4, IP6, IPX, OSPF
Network-Access layer	Ethernet, Token Ring, FDDI, HDLC, IEEE 802.11, IEEE 802.16, VLAN, X.25, ARP, RARP

Transmission Control Protocol (TCP)

The Transmission Control Protocol (TCP) is a connection-oriented protocol that makes use of the Internet Protocol as a transport protocol. This protocol establishes an acknowledged link session before sending any data. TCP offers a number of benefits, including reliability, flow control, sequencing, and error detection and correction. The link is closed until the data transmission is completed. If the source host does not receive an acknowledgment of receipt within a certain amount of time, it retransmits the data. This is referred to as a TCP link timeout [40].

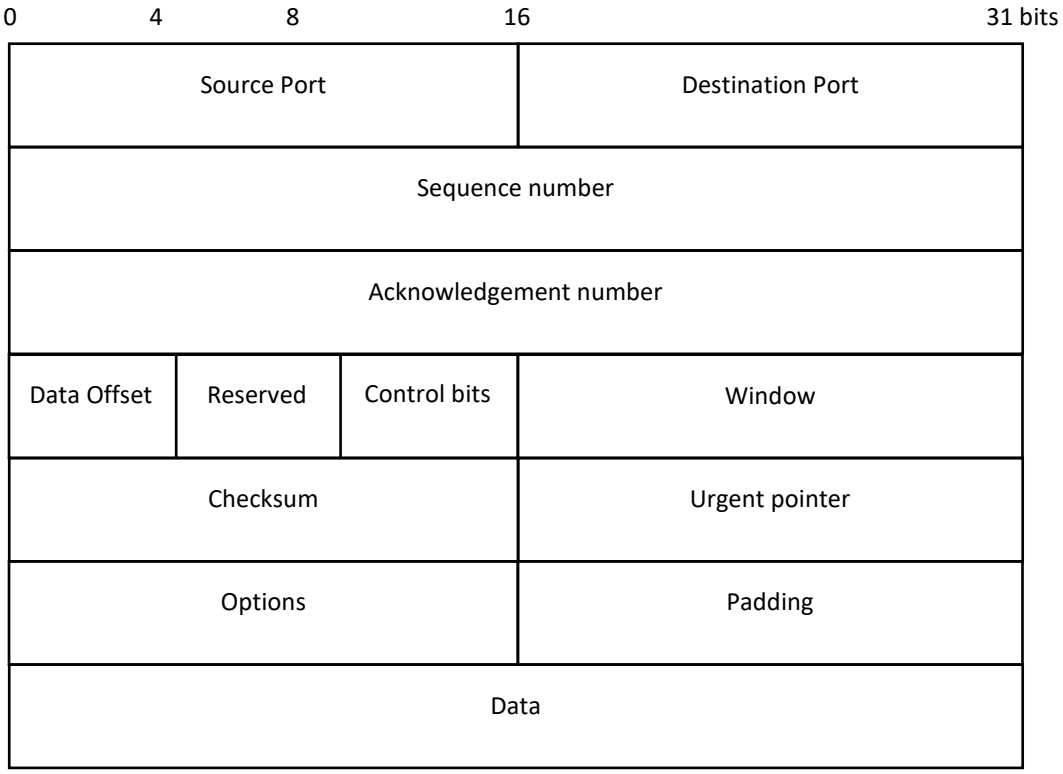


Figure 6: TCP Header Format

The Transmission Control Protocol header (Figure 6) or the TCP header contains the parameters and state of an end-to-end TCP socket. A socket is a software framework within a computer network's network node that serves as an endpoint for sending and receiving data. The TCP header is used to keep track of how communication between two TCP endpoints is progressing.

Internet Protocol (IP)

Data is transmitted from one device to another using the Internet Protocol. It is a connectionless protocol, which means that delivery cannot be guaranteed. It fragments and reassembles IP packets at the source and destination ends, respectively, to ensure delivery. Because of the existence of a maximum transmission unit (MTU) size for transmission, fragmentation is necessary. IP addressing is another essential IP activity

[40]. IP header, which is depicted below consist with header information at the beginning of an Internet Protocol (IP) packet [40]. Figure 7, illustrate the components of IP headers.

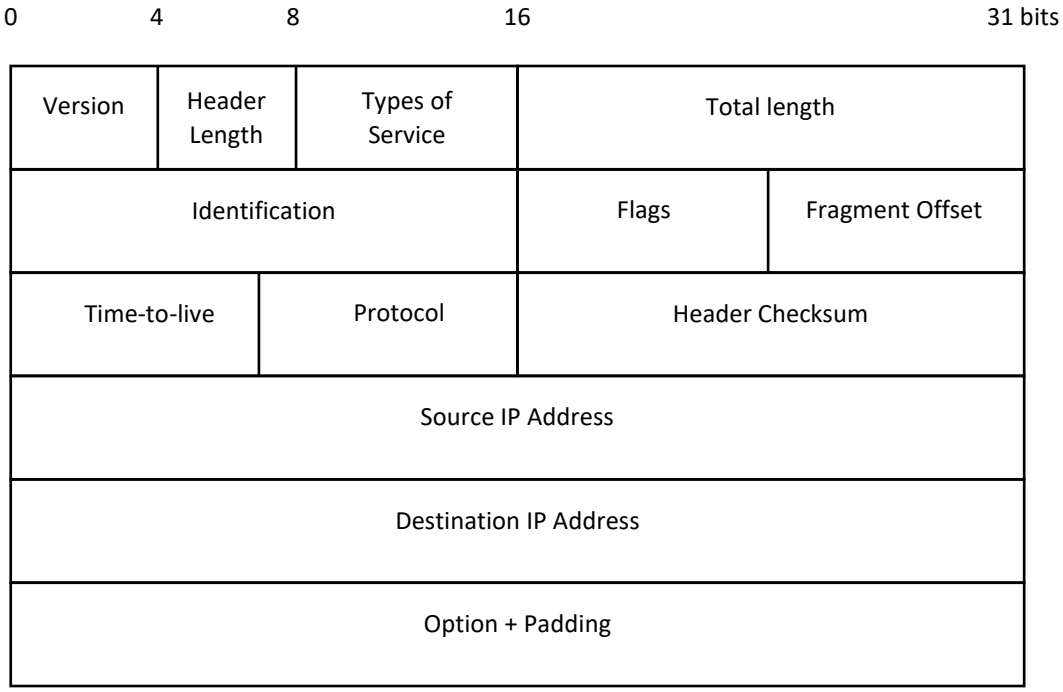


Figure 7: IP Header Format

The bulk of the research of network-based intrusion detection systems is performed at the application layer. NIDS often examine behaviour at the transport and network layers, both to detect attacks and to aid in the analysis of activity at the application layer (e.g., a TCP port number may indicate which application is being used). Some network-based Intrusion Detection and Prevention Systems often perform restricted hardware-layer analysis [23].

2.1.2 Intrusion Detection Systems (IDS)

Intrusions are attempts to undermine the confidentiality, integrity, and availability of computer networks. Intrusion detection is the method of detecting and reacting to certain aggressive activities. Intrusions detecting tools can build as a software-based or as a hardware-based solutions. Intrusion detection systems (IDS) play a significant role in protecting computer networks. IDSs are in charge of scanning computer network dataflows for violations of Cybersecurity policies and procedures. IDS alone is inadequate to secure an organisation's network.

Firewall

A firewall (Figure 8) is a network security device, which configured between the internal network and the external network. Its purpose is to protect the internal network from external threats. Furthermore, it controls the network traffic by blocking and granting access to the internal network on specified ports as per to pre-defined set of rules [41] .

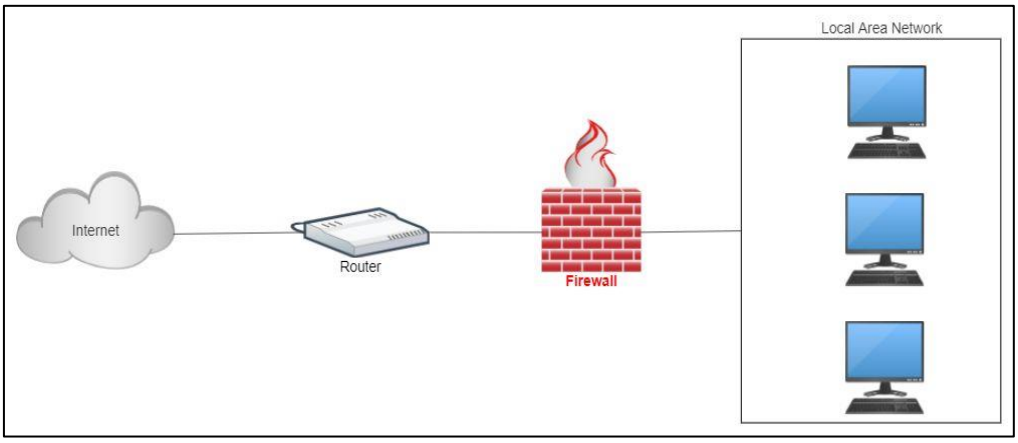


Figure 8: Firewall Example

Firewalls are available in both hardware and software configurations. The ability of firewalls to detect and prevent threats is limited by pre-defined configuration settings and guidelines. As a result, a second line of defence must be included, capable of detecting early warning signs of threats and reacting fast enough before the harm is done [42]. The underlying concept of systems such as Intrusion Prevention Systems (IPS) and Intrusion Detection System (IDS) is to identify and analyse threats and react proactively before Cyber adversaries cause damage to the organisational network. Iqbal et al. presents a comparison between firewall and IDS [43]. Refer Table 5 for more details.

Table 5: Difference Between IDS and Firewalls

Firewall	**IDS**
Device or application	Device or application
Analyses packet headers	Analyses whole packets
Enforces policies	Looking for known events
Block traffic based on network information	Monitor's traffic
Screening network traffic	An IDS monitors traffic
Monitors the system based on the rules	Monitors the system for unwanted entries

Intrusion Prevention Systems (IPS)

Intrusion prevention is the method of identifying potential intrusions and trying to prevent them from happening [23]. Before the suspicious event hits the mark, IPS will dynamically block it. IDSs, on the other hand, only provide warnings on possible security threats by gathering data from the computing environment and using detection algorithms to separate potential attacks from normal activities [44]. An example of a layout for IPS depicted in Figure 9.

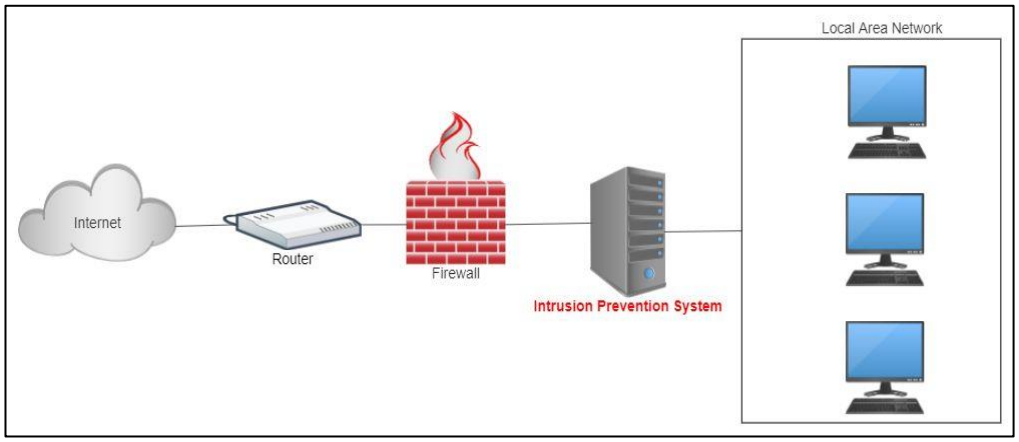


Figure 9: Intrusion Prevention System Example

Most organisations implement both IDS and IPS as part of their network security framework to strengthen the defences against Cyber-attacks. Majority of IPS developers depend on Bayesian strategy to implement probabilistic based detection modules [45]. Refer Table 6 to the distinct characteristics of IDSs and IPSs.

Table 6: IDS vs. IPS in Nutshell

	IDS	**IPS**
Name	Intrusion Detection System	Intrusion Prevention System
Narrative	IDS monitor the network traffic for dubious activity and alerts the users when such activities are discovered	Like IDS, IPS monitor the network traffic and alerts users in the event of suspicious activity. Also, the system takes preventive actions against the discovers suspicious activities
Placement	Host-based IDSs are installed in the client computers. Network-based IDSs reside on the network	Located between the firewall and the internal network
Use	Warns about the suspicious activities	Warns about suspicious activity and take preventive actions against them
Practical Implications on false-positive detection	False-positive detection has a minor effect. IDS only flag the concerns and does not block the traffic from entering the internal network	False-positive detection may consider legitimate network traffic as suspicious and block them from entering the internal network

It is more of an art than a science to distinguish network intrusions from regular traffic. SOCs are critical for capturing, normalising, storing, and correlating events in order to

detect malicious activities in an organisation's security infrastructure [46]. Log details from network devices such as firewalls, Intrusion Detection Systems (IDS), proxy servers, application servers, DNS infrastructures, and so on are examples of security incidents. [47]. In their academic work, Akinrolabu et al. highlight three major themes (Figure 10) for enhancing SOC operations. These three main themes, namely people (analysts), operations (processes), and technology (tools), are developed through industry interviews.

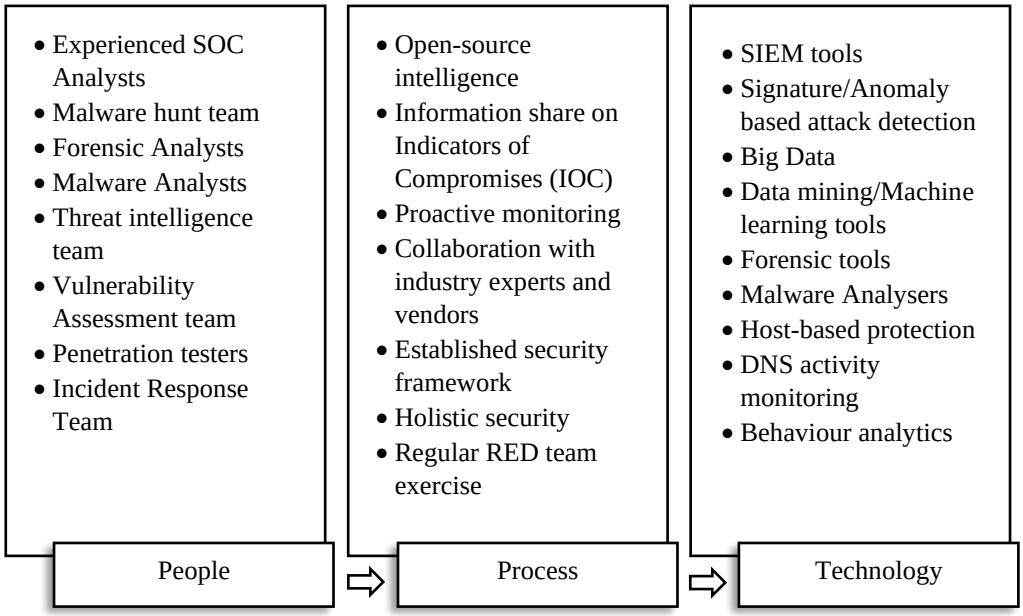


Figure 10:Three Major Themes for Improving SOC Operations *[47]*

As depicted above, improving the technologies related to attack detection and host-based protection play an integral part in securing organisational IT infrastructures. IDSs are responsible of detecting and protecting internal network from Cyber-adversaries. Since intrusion detection is so critical, researchers and industry have built and developed a variety of intrusion detection systems (IDSs) [48]. IDSs can be categorised based on their system properties and individual characteristics Table 7.

Table 7: Categorisation of Intrusion Detection Systems

Category	IDS Type	Description
Threat detection sensor	Software solution	Software-based IDSs are capable of rapidly adapting to unknown attacks (zero-day attacks). However, as compared to hardware IDS solutions, they have a slower detection speed [49].
	Hardware solution	Hardware-based IDSs have a faster detection rate than software-based IDSs, but they are ineffective against unknown unknown attacks. This solution requires physical equipment [49].
Architecture/ Data source	Host-based	Host-based intrusion detection systems (IDS) are installed on the server system to monitor and audit the information received from the operating system in order to detect intrusions. At the transport layer, these IDSs will deal with the encrypted world [50].
	Network-based	These are hardware/software devices that are installed at various key points in a communication network system, such as routers, switches, and gateways, and which continuously track and analyse all traffic flowing in and out of the network in order to identify suspicious activities [51].
	Hybrid	An IDS that is dependent on both the host and the network [52].
Attack detection technology	Misused-based	Signature-based IDS is another name for it. This IDS compares device and/or network operations to a list of established attack signatures; as a result, it is unable to detect zero-day attacks [48].
	Anomaly-based	The process of identifying significant deviations by comparing definitions of what constitutes normal activity with observed events [23].
	Hybrid	An intrusion detection system that employs both misuse-based and anomaly-based attack detection techniques [53].
Deployment architecture	Non-distributed	A non-compound IDS that can only be used in one location [48].
	Distributed	A compound intrusion detection system (IDS) is made up of several intrusion detection subsystems that can be deployed in different locations and interact to share intrusion detection-related data such as attack warnings. Distributed IDSs are capable of detecting orchestrated attacks that target multiple sites in a particular time order. [48].

2.1.3 Threat Detection Architectures

Let us discuss more about intrusion detecting platforms. Azeez et al. have conducted an analysis on different IDS architectures [54]. They have identified following IDS types:

Host-based Detection (HIDS)

HIDS investigates system-specific settings including software calls and local security protocols, as well as auditing host logs for suspicious behaviour. Each detection platform has their own pros and cons. Considering the characteristics of HIDS Azeez et al. have listed the plus points and the negative points of HIDS.

Advantages of HIDS

- It has the ability to detect intrusions into host applications, operating systems, and network layer traffic
- It can track and investigate suspicious activity over encrypted communication
- It can track the file system, file access, system calls, and other aspects of the host system to detect intrusion
- Since it is mounted on the host machine, it does not require any external hardware
- Since it communicates with both the user and the server-installed program, it can detect profile misuse
- It can detect attacks that NIDPS cannot detect and prevent intrusion at the device level

Disadvantages of HIDS

- Since it does not rely on a predefined database, detection accuracy is restricted

- It consumes more host resources, resulting in a decrease in device host efficiency
- It must be mounted on each host that will be monitored
- Its monitoring is limited to the host it is mounted on
- There is a chance it will clash with your current security settings

Network-based Detection (NIDS)

The Open System Interconnection (OSI) model's network, transport, and application layers are all covered by NIDS technology. Sensors, one or more management servers, several consoles, and one or more database servers are usually included in NIDS (if the network-based IDPS supports their use). Except for the sensors, all of these components are identical to those used in other forms of IDS architectures. An IDS sensor that tracks and analyses network activity on one or more network segments is known as a network-based IDS sensor. In addition, network interface cards that will be used for monitoring are set to promiscuous mode. This mode accepts all incoming packets, regardless of their intended destination. The majority of IDS deployments employ numerous sensors, with massive IDS deployments employing hundreds. Sensors come in both software and hardware versions [23].

HIDS and NIDS can be considered as the most popular schemes used in the field of IDSs. Table 8 shows a summary of comparisons between HIDS and NIDS [16].

Table 8: Comparison of HIDS and NIDS Technologies

Tech.	Advantages	Disadvantages	Data source
HIDS	End-to-end encrypted communications behaviour can be tested with HIDS.	Attacks are not recorded quickly enough.	Records, log files, the Application Programming Interface (API), rule patterns, and system calls are all audited.
	No additional hardware is needed	Consumes the assets of the host	
	Tests the host's file machine, system calls, and network events for intrusions	Each host must have it installed	
	Any packet is disassembled and reassembled. Looks at	It can only keep track of attacks on the HOST	

	the whole item, not just the streams		
	Looks at the whole item, not just the streams	It can only keep track of attacks on the computer where it is installed	
NIDS	Checks network packets to detect attacks	The problem is detecting attacks in encrypted traffic	• Simple Network Management Protocol (SNMP) • Network packets (TCP/User Datagram Protocol/Internet Control Message Protocol) • Management Information Base (MIB) • Router NetFlow records
	It is not necessary to install on each host	It is important to have dedicated hardware	
	Several hosts can be verified at the same time	It supports only identification of network attacks	
	Capable of detecting a wide variety of network protocols	Analysis of a high-speed network is difficult	
		Insider attacks are the most serious threat	

Apart from the main two variants, Azeez et al. have listed two other monitoring technologies [54]. Namely, Wireless (WIDS) and Network behaviour analysis (NBA)

Wireless Intrusion Detection (WIDS)

WIDS is a variant of NIDS which monitors and analyses packets and protocols on a wireless network.

Advantages of WIDS

- It works well for tracking and analysing wireless network intrusion.
- WIDPS can detect a range of problems at the wireless local area network protocol level, such as code breaches and misconfigurations.

Disadvantages of WIDS

- It is vulnerable to denial-of-service attacks.
- It is unable to track and analyse packets at the transport, network, and device layers.

- If an attacker attacks a channel that is not being tracked, it is open to evasion techniques.

Network Behaviour Analysis (NBA)

NBA is a type of NIDS that can track and analyse network traffic in order to detect suspicious activities such as policy violations, DDoS attacks, or malware intrusion.

Advantages of NBA

- It is good at detecting DoS attacks.
- It can be used to track packets on the transport network, network application TCP/IP layer, and so on.
- It can track and detect malware, policy violations, and DDoS attacks.

Disadvantages of NBA

- Packets are analysed in batches, which slows down intrusion detection.

2.1.4 Threat Detection Metrices for IDSs

IDSs distinguish abnormal events from daily network traffic in a number of ways. In reality, if an event is from a distribution of intruder activity events or from a valid user distribution, the intrusion detection output is defined [13]. IDS may or may not signal a warning during event processing. There are a number of classification metricises for IDS, some of which go by different names. The Figure 11 shows the IDS event outcomes categorisation uncertainty matrix, which can be used to determine an IDS's efficiency. The matrix's columns represent network traffic instances, while each row represents a response process. Detecting False Positives and False Negatives can be categorised as erroneous detections. Ideal IDS solutions should have minimum FP & FN detection rates.

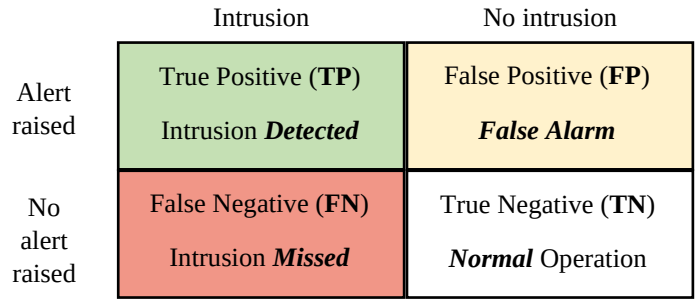


Figure 11: IDS Alert Categorisation

True Positive

When the device senses an anomaly or an intrusion, it should immediately alert the appropriate authorities. The True Positive Rate of an ideal IDS should be higher (TPR). The ratio between the number of correctly expected attacks and the total number of attacks is used to measure TPR. TPR is also known as Detection Rate (DR) Recall or IDS Sensitivity. Furthermore, developing such IDS is a challenging task [16]. In a nutshell this metric represents the attack detection capability of the model.

$$TPR = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

True Negative

The proportion of right benign classifications out of a given set of benign instances is known as specificity or true negative rate [22]. The True Negative Rate (TNR) is the proportion of correctly categorised Normal samples to the total number of Normal samples.

$$\text{TNR} = \frac{\text{True Negatives}}{\text{True Negatives} + \text{False Positives}} = 1 - False\ Positive\ Rate$$

False Positive

The ratio between the number of normal instances wrongly classified as an attack and the total number of normal instances is calculated as the False Positive Rate (FPR) [16]. False positives are a common problem with IDSs, and they cause human analysts to become distracted. Furthermore, FP detection produces a stream of false alarms, causing security operators to become overwhelmed. As a result, FPR strongly restricts an IDS's usability.

$$FPR = \frac{False\ Poitives}{False\ Positives + True\ Negatives}$$

False Negative

False negative rates, also known as miss rates, are a security flaw that can result in unknown harm if intrusions are missed. A false negative suggests a problem with the detection of malicious behaviour. The False Negative Rate (FNR) is the rate at which an anomaly is detected and labeled as natural.

$$FNR = \frac{False\ Negatives}{False\ Negatives + True\ Positives} = 1 - True\ Positive\ Rates$$

As previously reported, a false positive is an incorrect identification of benign traffic, whereas a false negative is a failure to detect malicious behaviour. The number of benign and malicious flows will help us figure out how efficient an algorithm is [49].

Detection Accuracy

The accuracy or classification rate (CR) is defined as the percentage of correctly predicted instances to all instances. This metric assesses how well the IDS senses normal and irregular traffic patterns [16].

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Positive Predictive Value

The proportion of accurate classifications out of a set of predictions categorised as attacks is known as the Positive Predictive Value (PPV) or Warning Precision (AR) [22].

$$AP = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

It is difficult to eradicate both false positive and false negative detections. However, we can monitor the situation by reducing the frequency of one while increasing the frequency of the other. Many companies prefer to eliminate false negatives at the expense of false positives, resulting in more malicious events being reported but more research resources being required to separate false positives from actual malicious events. Tuning is the method of modifying an IDS's configuration to boost its detection accuracy [23].

P. C. Van Oorschot have provided a decent example to understand the practical usage of above matrices [13]: Consider the following situation. There is an infection X, and a test that screens for it. Provided 100 healthy individuals, the test detects one diseased

person on average, resulting in one false positive, and FPR = 1/(1+99) = 0.01 = 1%. As a result, TNR = 1 and FPR = 99 percent. Also, given 100 diseased people, the test finds 98 diseased people on average, resulting in two false negatives, and FNR = 2/(98+ 2) = 0.02 = 2%, or TPR = 98/(98 + 2) = 0.98 = 98 percent (Such a test may be sold as "98% accurate" or "99% accurate," but failing to clarify the measure used would confuse experts and non-experts alike; we will come back to this point later.) Assume that the prevalence of our disease X in the general population is 1 in 100,000; in a random group of 100,000 individuals, we should expect 1 to be afflicted. If the screening test is used on this package, we can expect to find 1% of 99,999, or 1000 false positives (based on the 1% FPR). The one diseased person would almost definitely test positive as well (due to the 98 percent TPR). Of course, the doctors see a result of 1001 people being flagged as "maybe having disease X," so 1000 of the 1001 are false alarms. We can now believe we were fooled solely by the earlier metrics, as well as any claims that the test was 98 percent or 99 percent “accurate” Alarm precision (AP) is a new metric that measures the proportion of correctly raised alarms to total alarms (true positives to total positives):

$$AP = TP/(TP + FP) = 1/(1 + 1000) = 1/1001 \approx 0.1\%$$

Positioning this in reverse as alarm imprecision, AIP=FP/(TP+FP) = 1000/ (1+1000) ≈ 0.999. We now see that 99.9% of alarms raised are false alarms.[1] An IDS with a high ratio (high AIP) and a high absolute number of false alarms has a problem.

In this case, we are being misled by missing the disease's low base rate of incidence across the population. This is the proportion of intrusion events to overall events for an IDS. For our clarification, we turn to an IDS setting: "diseased" becomes an intrusion case, and "positive test result" becomes an IDS alarm raised. Let us go over the above example algebraically again, this time using approximations that apply to that case—where the number of false positives far outnumbers the number of true

[1]FPR = 1000/1001 measures false alarms over all events involving positive samples, while AIP = 1000/1001 measures false positives over all events involving no disease (intrusions) (alarms).

positives., i.e., TP << FP (a difficult situation for an IDS). Assume a set of *n* events. It consists of n_I intrusion and n_N non-intrusion events (n is known, but not n_I, n_N).

$$n = n_I + n_N, \qquad TP = TPR \cdot n_I, \qquad FP = FPR \cdot n_N$$

The last two equations are just the definitions. To a rough approximation, we expect an IDS to detect a high proportion of intrusions, TPR ≈ 1 implying TP ≈ n_I. A tiny base rate of intrusions (as also expected in a fielded IDS) means $n_I << n_N$ and $n \approx n_N$. From $TP << FP$ we get $TP + FP \approx FP$. Now substituting in these approximations,

$$AP = TP/(TP + FP) \approx TP/FP \approx n_I/(FPR \cdot n_N) \approx (n_I/n)/FPR$$

This approximation for AP now captures the parameters that dominated the computed value of AP in our example: the base rate n_I/n of incidence, and FPR. In summary, both the base rate of occurrence and the false positive rate influence warning accuracy.

2.2 Cybersecurity Threats and Threat Detection

According to the CyberEdge group's 2020 Cyberthreat Defence Report (Figure 12) , the percentage of organizations impacted by an effective Cyber-attack increased to 80.7 percent, up from 78.0 percent last year. DoS attacks, Malware, Phishing / spear-phishing attacks, Ransomware, Account takeover / password misuse attacks, and Zero-day attacks are among the above attacks [55].

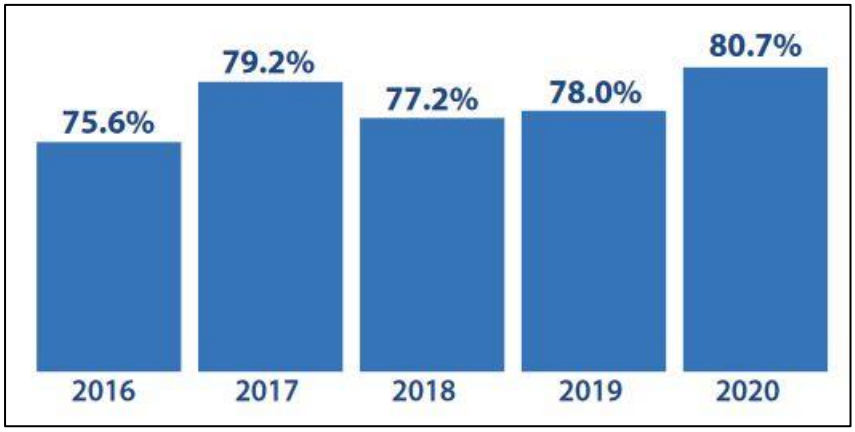


Figure 12: Percentage Compromised by at Least One Successful Attack, by Year

The findings of Ipsos Mori's 2020 Cybersecurity Breaches Survey for the UK Government's Department for Digital, Culture, Media & Sport (DCMS) revealed that Cyber-attacks are becoming more complex and frequent [56]. Moreover, in the United Kingdom almost half of businesses (46%) and a quarter of charities (26%) report having Cybersecurity breaches or attacks in the last 12 months. Like previous years, this is higher among medium businesses (68%), large businesses (75%) and high-income charities (57%) [57]. Cybersecurity-related incidents reported to Sri Lanka CERT increased in 2019 compared to the previous year, according to the 2019 Annual Activity Report of the Sri Lanka Computer Emergency Readiness Team. [58].

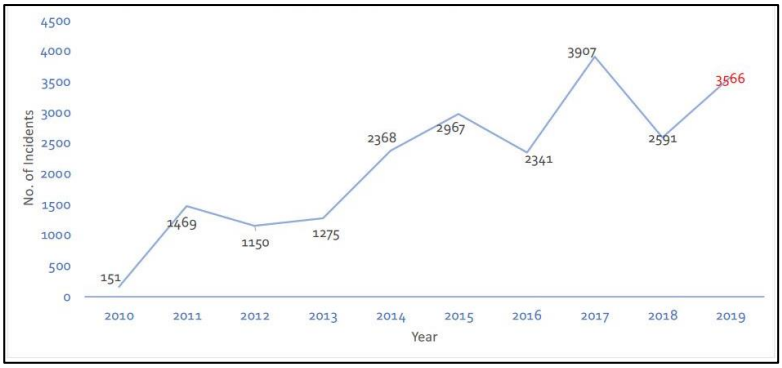


Figure 13: Number of Cyber Security Incidents Report to SL CERT from 2010 - 2019

Incident Type	Year									
	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019
Phishing	6	6	8	8	12	14	23	42	12	5
Abuse/Hate/Privacy Violation	20	2	8	8	8	21	32	29	11	307
Scams	10	3	6	18	12	18	12	32	7	5
Malicious Software/Ransomware	5	1	2	2	3	12	21	39	19	14
Financial Frauds	-	-	-	-	-	10	16	35	21	7
Web site Compromise	8	20	15	16	56	20	10	25	9	175
Compromised emails	12	3	6	8	10	16	16	14	-	21
Intellectual Property violation	-	5	3	3	3	3	7	6	6	1
Unauthorized Access	10	3	1	11	8	-	-	0	-	
DoS/ DDoS	-	1	1	1	6	3	4	0	1	2
Social Media Incidents	80	1425	1100	1200	2250	2850	2200	3685	2505	2662
Phone Hacking	-	-	-	-	-	-	-	-	-	1
Server Compromised	-	-	-	-	-	-	-	-	-	2
Other	-	-	-	-	-	-	-	-	-	364
Total	**151**	**1469**	**1150**	**1275**	**2368**	**2967**	**2341**	**3907**	**2591**	**3566**

Figure 14: Growth of the Types of Cybersecurity Incidents in Sri Lanka

Above statistics (Figure 13 and Figure 14) clearly shows that Cyber threats are rising at a rapid pace and securing IT infrastructures has become a crucial task to achieve. IDSs play a critical role in protecting systems against Cyber-attacks, but the increasing sophistication of Cyber-attacks presents a serious challenge to IDSs [16].

Based on their activities and motives, network threats or Cyber-attacks can be divided into four groups [59]:

- Denial-of-Service (DoS) attacks- The aim of a DoS attack is to prevent or limit the transmission of services over a network or computer device.
- Remote-to-Local(R2L)- The aim of this attack is to send packets to the victim computer.
- Probing attacks- These attacks collect information about the network or computer system.
- User-to-Root (U2R) attacks- The aim of this attack is for a non-privileged user to obtain root or admin-user access to a device or machine where the attacker had user-level access.

2.2.1 Anomaly Detection Techniques

In today's computer systems, intrusion detection systems (IDSs) are critical. As a result, they have been the object of adversarial intrusions [60]. IDSs' main job is to find various forms of malicious network traffic and device use that a typical firewall cannot detect [16]. The ability of IDSs to identify threats can be divided into two categories: signature-based detection and anomaly-based detection.

Signature-based Intrusion Detection (SIDS)

Pattern matching methods are used by SIDSs to detect known attacks. Matching techniques are used in SIDS to identify a prior intrusion. In other words, a warning signal is activated when an intrusion signature matches the signature of a previous intrusion that already exists in the signature database [16]. Knowledge-based detection systems or Misuse detection are other names for this technique. Figure 15 shows a high-level diagram of a SIDS.

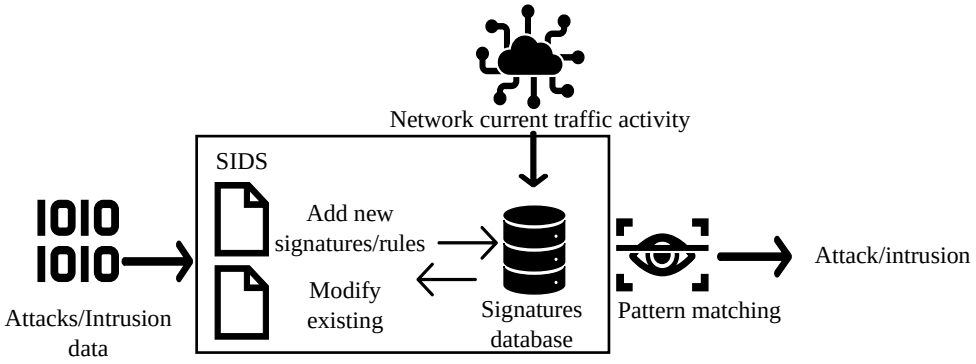


Figure 15: Signature-based Detection

Anomaly-based Intrusion Detection (AIDS)

In AIDS, usual behaviour of a computer system is created using machine learning, statistical-based or knowledge-based methods. An anomaly is characterised as a substantial difference between observed behaviour and the model, which can be interpreted as an intrusion. This collection of strategies is based on the premise that malicious behaviour varies from normal user behaviour. The training phase and the testing phase are the two stages in the development of AIDS. The normal traffic profile is used in the training phase to learn a model of normal behaviour, and a new data set is used in the testing phase to determine the system's ability to generalise to previously unknown intrusions. Based on the training process, AIDS can be divided into many groups. They are focused on statistics, intelligence, and machine learning. [16]. Fernandes Jr. et al. have presented few more methods, that can be used to train AIDSs through a thorough literature study [4]. Statistical methods, clustering methods, finite state machine methods, classification methods, information theory, evolutionary computation, and hybrid techniques are some of the techniques available. These strategies each have their own collection of features, benefits, and drawbacks.

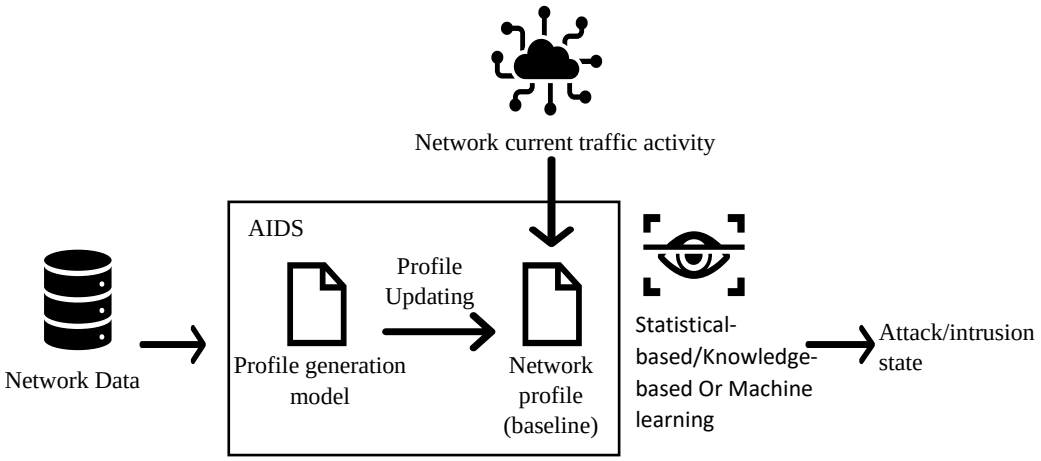


Figure 16: Anomaly-based Detection

The most popular IDS is anomaly-based detection, which is complex and detects both known and unknown anomalies. High-level diagram of an AIDS shown in Figure 16.

Table 9 show the advantages and limitations of Statistical-based training approach [4].

Table 9: Advantageous and Limitations of the Statistical Based Approach

Pros	Cons
More than any other tool, it has the inherent ability to detect network anomalies,	Some attacks may be a regular part of the training data collection, and they may be integrated into daily actions, making it normal
Ability to learn the expected behaviour of the traffic (network system),	To be able to set the first alarm, the models must be trained for a period of time.
Traffic analysis is based on the theory of sudden changes, which sets an alarm whenever a significant deviation happens.	Because of its restricted and static existence, the use of thresholds may not be reliable in certain real-world situations
As an input, the methods do not require any previous knowledge of the system.	

Advantageous and limitation of the clustering method (Table 10).

Table 10: Advantageous and Limitations of the Clustering Approach

Pros	Cons
Incremental clustering generates a fast response.	They rely heavily on proximity measures, and each one can have a positive or negative impact on the detection rate.
The performance is consistent as compared to statistical approaches or classifiers	It takes a long time.
Computational complexity is minimised due to the ability to separate large data sets into smaller ones.	They are not designed to detect anomalies.
	Algorithms can get stuck in local minima at times.

Techniques based on finite state machines are rarely used as statistical methods or classification techniques. However, there are some advantages of using this approach. (Table 11).

Table 11: Advantageous and Limitations of the Finite State Machine Approach

Pros	Cons
Flexibility and robustness	Time-consuming.
Because of their alphabet of input and output, strong analytical techniques can reflect a broad range of circumstances	Inability to detect attacks that are unusual or indefinite
A high degree of detection if there is a significant knowledge base about attacks and normal circumstances	The cost of dynamically updating rules/conditions is high

Classification based techniques is one of the machines learning detection approach that can be used to detect anomalies. Classification based approach is commonly used to detect threats in IDSs. Fernandes Jr. et al. have further classified this method into several other techniques. Naïve Bayesian, Support vector machines, Artificial neural network, Ensemble approach can be considered as few notable mentions. Advantages and disadvantageous of this classification listed down in Table 12.

Table 12: Advantageous and Limitations of the Classification Based Approach

Pros	Cons
Elasticity for assessing and training by integrating new information into the execution strategies	High use of resources
High detection rates for attacks that have been confirmed	Without any relevant training information, the inability to detect unknown anomalies
Artificial Neural Networks are adaptable, allowing for gradual training and testing of cases	The use of neural networks can result in over-fitting
Multi-level neural network techniques outperform single-level neural network techniques in terms of performance	For large data sets, selecting sample data sets is time consuming
By combining several classifiers, even if they are weak, ensemble methods work well	Real-time results can be difficult to obtain in some situations

To date, machine learning-based threat detection has had a good track record. Furthermore, most machine learning models have used approaches such as manual network inspection, network metadata, or structural similarities to pick features. Unfortunately, the majority of this performance is highly dependent on the network context in which the algorithms were developed and the data set that was used [61]. Furthermore, most machine learning models have used approaches such as manual network inspection, network metadata, or structural similarities to pick features.

Information Theory is a mathematical topic that focuses on the quantification of data and the study of redundancy. Information theory-based detection methods such as entropy and Kullback-Leibler distance can be considered as good examples. Refer to Table 13 for pros and cons of Information theoretic-based approach.

Table 13: Advantageous and Limitations of the Information Theoretic-based Approach

Pros	Cons
It is possible to operate in an unsupervised mode	The use of statistical properties for time series of traffic-related features (e.g., Gaussian) can lead to errors
The primary statistical distribution of the data is not assumed in any way	Anomalies may only be detectable if there is a large presence of them in the data collection. In this case, a highly sensitive information theoretic measure is needed to detect anomalies caused by a small number of anomalous patterns
The complexity of time and space is a minor issue for information theory-based approaches since they only use header information for calculation	It is difficult to connect an anomaly score to a trial event.

Bioinspired Computing or the Evolutionary Computation encompassed with methods such as Artificial Immune System (AIS), Genetic Algorithms (GA), Differential evolution, Particle swarm optimisation. Advantageous and Limitations of the Evolutionary computation approach are shown in Table 14.

Table 14: Advantageous and Limitations of the Evolutionary Computation Approach

Pros	Cons
They improve the parallel processing capabilities of intrusion detection systems	Finding the fitness function could not be easy
It is not necessary to have prior knowledge of the problem domain	Choosing the best parameters is difficult
The ability to naturally retrain makes the whole system more adaptable	It can be difficult to translate an issue into a biological approach at times.
The presence of noise and discontinuities in the data set has no significant effect on the solutions	

To improve anomaly detection capabilities, the hybrid approach incorporates different groups of algorithms, approaches, and techniques. Advantageous and Limitations of the Hybrid anomaly detection approach are listed in Table 15.

Table 15: Advantageous and Limitations of the Hybrid Anomaly Detection Approach

Pros	Cons
Hybrid methods may benefit from both anomaly and signature-based approaches' key features	When more methods are combined, there is a greater demand for computational resources, which raises the cost
They are capable of detecting both well-known and unknown anomalies.	The topic of dynamism is still unresolved

Table 16 present the pros and cons of SIDS and AIDS while

Table 17 shows the distinct Characteristics of SIDS and AIDS

Table 16: Comparison of Intrusion Detection Techniques

	Pros	Cons
SIDS	Highly successful at detecting intrusions with the fewest possible false alarms (FA)	A new signature must be added on a regular basis
	Identifies intrusions quickly	SIDS is a security system that detects attacks based on known signatures. If a previous intrusion is slightly changed into a new version, the device is unable to detect this new variant of a similar attack
	Superior in terms of detecting identified attacks	The zero-day assaults are undetectable
	Simple design	Detection of multi-step attacks is not possible
		There is a lack of comprehension of the attacks' insight
AIDS	Able to spot new attacks.	Since AIDS is unable to handle encrypted packets, the assault will go undetected and pose a danger
	It is possible to use AIDS to make an intrusion signature	A high number of false positive alarms
		It is difficult to create a standard profile for a highly complex computer system
		Consists of unclassified alerts.
		Requires initial training

Table 17: Characteristics of SIDS and AIDS

	SIDS	AIDS
Detection performance	Low false alarm rate: incapable of detecting unknown attacks and necessitates the maintenance of a broad signature database	Unknown (zero-day) attacks are possible; there is a high incidence of false alarms
Detection efficiency	High, but decreasing as the size of the signature database increases	Depending on the sophistication of the model
Dependence on domain knowledge	Almost every detection relies on prior knowledge of the domain	Low, domain knowledge is only needed for feature design

Interpretation	Domain knowledge-based design, high interpretive capability	Only produces detection results and has a limited capacity to interpret them
Unknown attack detection	Only known attacks are observed	Detects both known and unknown threats (Zero-day)

Though there are multitudes of ways to build a IDSs, the correct identification of the requirements, technology, resources are crucial to the development of an IDS. In fact, having the more technically advance IDSs does not assure the protection of networks and computer systems unless it aligns with the user requirements.

2.2.2 Machine Learning and Intrusion Detection

Machine learning and related technologies have a big effect on intrusion detection and Cyber-attack detection. An introduction to machine learning, algorithms, and the relationship between machine learning techniques and intrusion detection will be explored in this section.

Machine Learning

Artificial intelligence(AI) includes machine learning as a subset Figure 17. Artificial intelligence refers to the ability to think or comprehend something that is not normal or produced by humans [38]. Machine learning aims to extract valuable knowledge from large amounts of data(big data) [62].

Machine learning is a powerful intrusion detection technique and tool, especially for anomaly detection [63]. With high accuracy, machine learning methods can automatically detect the essential differences between normal and abnormal data. Furthermore, since machine learning methods are highly generalisable, they can detect previously undetected attacks [64].

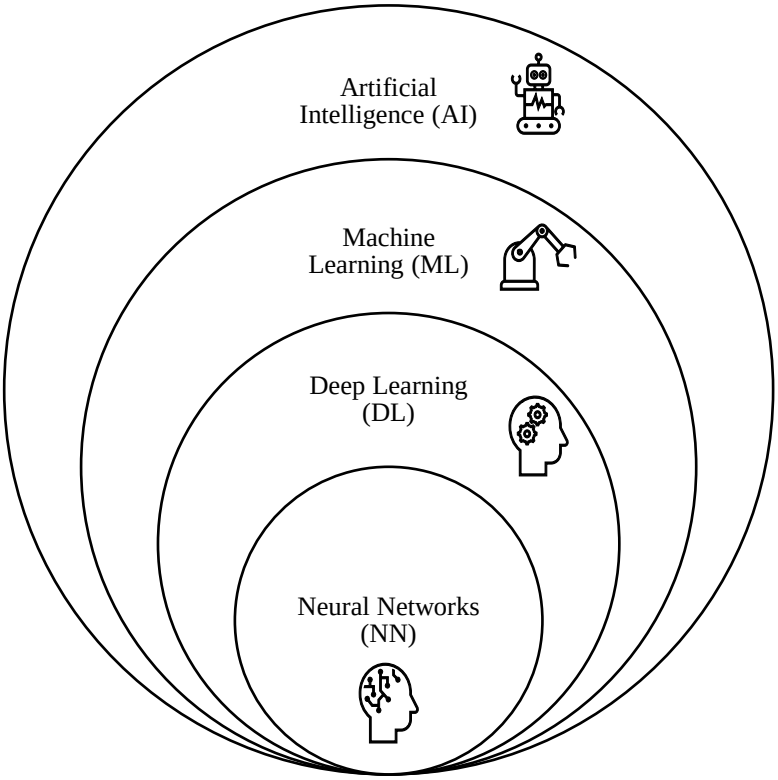


Figure 17: Classification of Different Learning

2.2.2.2 Common Machine Learning Algorithms in IDSs

Machine learning algorithms can be divided in two categories.

- Supervised Learning
- Unsupervised Learning

The success of supervised learning is dependent on the availability of useful information in labeled data. The most popular role in supervised learning (and also in IDS) is classification; however, manually labelling data is costly and time consuming. As a result, the key bottleneck to supervised learning is a lack of adequate labelled data [64].

Unsupervised learning, on the other hand, derives useful feature information from unlabelled data, making training data far easier to come by. Unsupervised learning

methods, on the other hand, usually perform worse in terms of identification than supervised learning methods [64].

Figure 18 depicts the most popular machine learning algorithms used in IDSs.

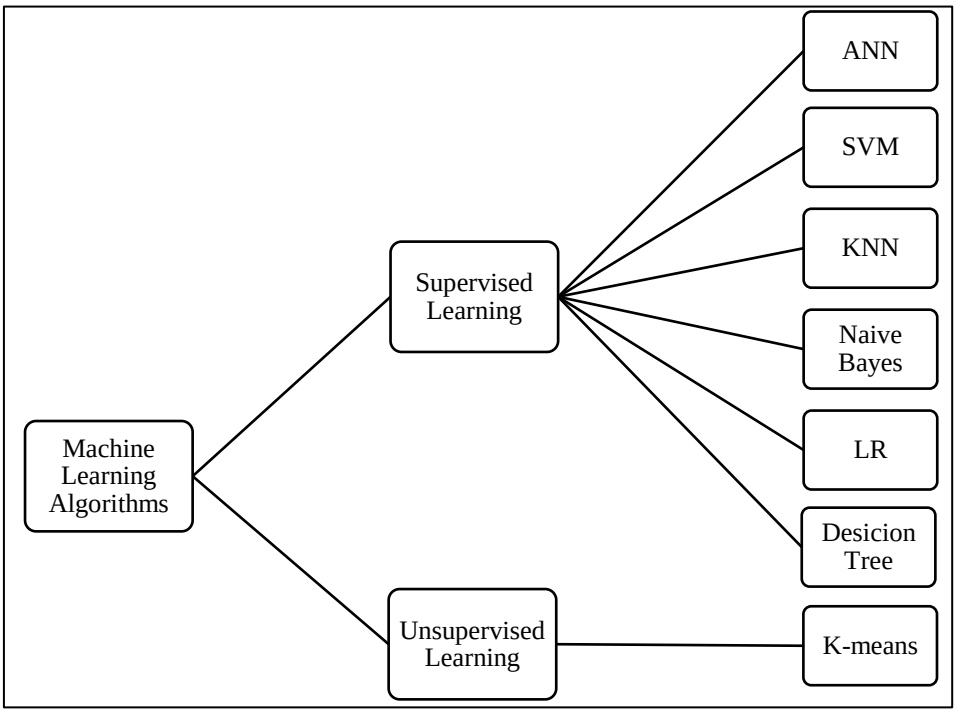


Figure 18: Common Machine Learning Algorithms

Artificial Neural Network (ANN)

An ANN is programmed to function in the same way that human brains do [64]. It is made up of neurons (nodes), which are processing components, and the connections between them. An input layer, several hidden layers, and an output layer organise the nodes [62]. Pros and Cons of ANN are listed in Table 18.

Table 18: Pros and Cons of ANN

Pros	**Cons**
Capable of working with nonlinear data	Apt to overfitting
Strong fitting ability	Often gets trapped in a local optimum
	It takes a long time to train a model

Support Vector Machine (SVM)

In SVMs, the aim is to locate a maximum-margin separation hyperplane in the n-dimensional feature space. Since the separation hyperplane is defined by a small number of support vectors, SVMs can produce satisfying results even with small-scale training sets. SVMs, on the other hand, are susceptible to noise near the hyperplane [64]. By correctly predicting the regular and malicious groups, the SVM algorithm can be used to improve the efficiency and accuracy of NIDS [62]. Pros and Cons of SVM listed in Table 19.

Table 19: Pros and Cons of SVM

Pros	Cons
From a small train set, you will learn valuable knowledge	May not perform well in projects involving large amounts of data or multiple classifications
Capability to generate large amounts of energy	Kernel function parameters are sensitive

K-Nearest Neighbor (KNN)

The manifold theory is the foundation of KNN. If the majority of a sample's neighbours are from the same class, the sample is likely to be from that class as well. As a consequence, the classification outcome is limited to the top-k nearest neighbours [64]. The model can be prone to over-fitting if the value of k is very small. While a broad k value selection can result in misclassification of the sample case, researchers compared the output of different ML algorithms using the CSE-CIC-IDS2018 data set, which is up to date. They solved the data set imbalance problem by using Synthetic Minority Oversampling Technique (SMOTE) to reduce the imbalance ratio, which improved the detection rate for minority class attacks. [62]. Advantageous and shortcomings of ANN listed in Table 20.

Table 20: Pros and Cons of KNN

Pros	Cons
Apply to large amounts of data	The minority class has a low accuracy rate
Appropriate for nonlinear data	Longer times of research
Train as soon as possible	The parameter K has a strong impact on the sensitivity of the method
Noise-resistant	

Naïve Bayes

The conditional probability and the hypothesis of attribute independence are the foundations of the Naïve Bayes algorithm. The Naïve Bayes classifier calculates the conditional probabilities for various groups for each sample [64]. Benefits and inadequacies of Naïve Bayes classifiers are listed in Table 21.

Table 21: Pros and Cons of Naive Bayes

Pros	Cons
Noise-resistant	When it comes to attribute-related data, Naïve Bayes don does not do so well
Able to learn in small steps	

Logistic Regression (LR)

This is a type of logarithm linear model [64]. Logistic regression is a statistical model that uses a logistic function to model a binary dependent variable in its most simple form, though there are several more complex extensions available. Table 22 lists the advantageous and shortcomings of LR classification.

Table 22: Pros and Cons of LR

Pros	Cons
Simple and easy to learn	Nonlinear data does not work well
Scale features automatically	

Decision Tree

The decision tree algorithm uses a collection of rules to classify data. The model is tree-like, making it easy to understand. The decision tree algorithm can automatically filter out features that are irrelevant or redundant. Feature collection, tree generation, and tree pruning are all part of the learning process. When training a decision tree model, the algorithm chooses the best features one by one and creates child nodes from the root node [64]. Multiple decision trees are used in many advanced learning algorithms, such as Random Forest (RF)59 and XGBoost60 [62]. Advantageous and shortcomings of Decision Tree are listed in Table 23.

Table 23: Pros and Cons of Decision Tree

Pros	**Cons**
Features that are automatically selected	Classification outcome fads to majority class
Strong elaboration	Don does not pay attention to the data correlations

K-means

K-means is a popular clustering algorithm, with K denoting the number of clusters and means denoting the attribute mean. Distance is used as a similarity measure criterion in the K-means algorithm. The closer two data items are to each other, the more likely they are to be grouped together. Although the K-means algorithm adapts well to linear data, the results on nonconvex data are less than ideal. In addition, the initialization condition and the parameter K influence the K-means algorithm. As a result, several repeated experiments are needed to determine the proper parameter value [64]. Advantageous and shortcomings of this classifier are listed in Table 24.

Table 24: Pros and Cons of K-means

Pros	**Cons**
Simple and easy to learn	Nonconvex data does not work well
Exceptional scalability	Initialisation-sensitive
Can be used for large amounts of data	The parameter K has a strong impact on the sensitivity

Machine Learning in Threat Detection

Machine learning includes all methods and algorithms that allow machines to learn on their own using mathematical models to extract useful knowledge from large data sets [62]. Following figure (Figure 19) depicts a generalised model of integrating NIDS and Machine learning.

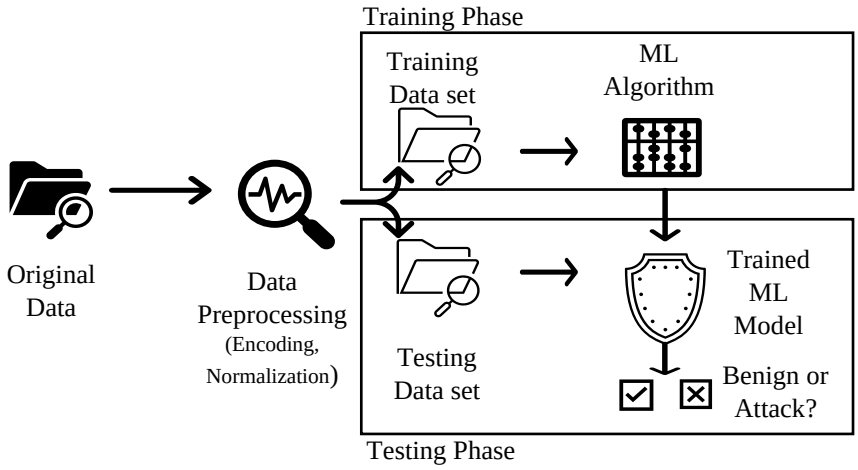


Figure 19: Integrating Machine Learning to a NIDS

IDSs' threat detection capabilities can be streamlined and improved using machine learning techniques. From 2010 to 2018, the rate of academic publications related to machine learning and Cybersecurity was clearly evident in the graph below [38]. As the amount of Machine Learning and Cybersecurity research grows over time, this is a booming research field. Figure 20 presents the number of papers published on ML and Cybersecurity from 2010 to 2018.

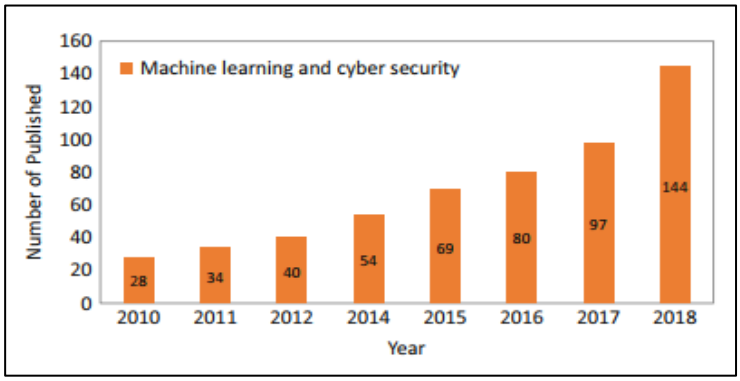


Figure 20: The Number of Papers Published on ML and Cybersecurity from 2010-2018

But there are "Lies, damned lies, and statistics", despite the staggering amount of research published each year, there is a remarkable discrepancy between the large amount of research on machine learning-based anomaly detection performed by academic researchers and the lack of operational implementation of such systems [61].

Sommer et al. established six domain-specific problems that make machine learning and anomaly detection functional implementation considerably more difficult. The following are some of the domain-specific challenges: [61]

- Outlier identification is important, but machine learning is better at identifying similarities.
- Classification errors have extremely high costs, making error rates seen in other domains unsustainable.
- There is a semantic difference between the detection results and the operational understanding of those results
- The immense variability of benign traffic makes finding predictable notions of normalcy challenging.
- There are a lot of barriers to completing a sound test
- The need to work in an adversarial environment.

It is important to emphasise that these considerations do not imply that machine learning is an ineffective method for intrusion detection. The authors conclude that the unfortunate combination of the aforementioned factors is at the root of the problem.

2.2.3 Feature Selection for Intrusion Detection

Reduce the computational complexity, minimise data duplication, improve the detection rate of machine learning techniques, simplify data, and reduce false alarms by using feature selection. Furthermore, Khraisat et al. divided feature selection strategies into two categories: (1) Wrapper method (2) Filter method [16].

Wrapper Method

This approach identifies possible interactions between variables by estimating subgroups of variables. There are a few drawbacks to this method: When the amount of data is insufficient, accumulative overfitting is used, and when the number of variables is large, important measurement time is used.

Filter Method

This method is frequently used as a pre-processing step. Some machine learning methods are not used in feature selection. Alternatively, features are nominated for their association with the outcome variable based on their scores in many statistical studies.

Since traditional machine learning models are unable to directly address flow data, feature engineering is needed before these models can be used. The “function vectors + shallow models” mode is used by feature engineering-based methods. Many machine learning algorithms may use the feature vectors. The semantics of each dimension of the feature vectors are easily understandable. The average packet length, the difference in packet length, the ratio of TCP to UDP, the proportion of TCP flags, and so on are all standard features. These types of detection methods have the

advantages of being easy to implement, highly effective, and able to meet real-time requirements [64].

The existing anomaly detection systems and software have limited capabilities. The lack of expert feedback from SOC analysts in the development of the tools limits their ability to detect novel attacks or variations of existing ones; the use of features that are templates aims to distinguish [47]. Akinrolabu, et al. has compiled a list of malware behavioural features (Table 25) based on the literature findings and based on the interviews they conducted with SOC analysts [47].

Table 25: Malware Behavioural Features Collated by Akinrolabu [44]

No.	Abbreviation	Feature Description	Class
1	Pkt_var	Strange file sizes - too large or too small (variance in size)	Web
2	TCP_NAck	More than three successive packets in the same direction without a return packet	TCP
3	Reg_Pkt	Same src/destination flow with same sized packets sent at regular intervals	TCP
4	Out_ SYN	OPEN SYN connections to 3 or more destinations without acknowledgement from servers for a duration of 3 seconds	TCP
5	Mal_Dst	The number of unpopular destination networks accessed by the same host	IP
6	DNS_Tun	Large size of request and response packets in domain name query exchanges	DNS
7	UDP_80	HTTP traffic over UDP port 80	Web
8	Encr_80	Encrypted, traffic on TCP/UDP ports 80	Wed
9	HTTP_Rfr	Missing Referrer and missing X-forwarded-for in HTTP request	Web
10	No_Resp	Zero HTTP response from the server.	Web
11	HTTP_Bin	HTTP GET request for *.exe (or POST containing config, binary or drop zone)	Web
12	HTTP_Ref	Uncommon HTTP referrer in user browsing history	Web
13	XOR_Enc	The use of XOR encryption for data obfuscation	Web
14	T_Urg	The number of urgent packets in the network traffic from and to a single host	TCP
15	Mal_IPs	Outbound traffic to IP addresses that belong to the same ASN as a set of known malicious IPs	IP
16	DNS_RR	Name server response to a domain query request, containing more than 5 IPs in the response.	DNS
17	New_Dom	Domain requests for a new domain whose age is less than one day old.	DNS

The following table lists features identified by the authors through their respective interview sessions (Table 26) with security analysts. The following features, according to Akinrolabu et al., will serve as the foundation for a malware detection model since

they are focused on malware trends and behaviours and are unaffected by malware structure [47].

Table 26: Identified Features by the Interviews (Akinrolabu) [44]

No.	Abbreviation	Feature Description	Class
1	HTTP_Rnd	Access to random URLs, with similar patterns at scheduled times of the day	Web
2	T_Pkts	The average time between packets in time interval	TCP
3	Login_FA	Consecutive failed login attempts before a successful login	Auth
4	Conn_Len	Long periodic connection (frequency 5s-60s), to a high port, with large packet sizes	TCP
5	ICMP_P	Strange pings (broad spectrum pings)	ICMP
6	PUSH_Reg	Nos. of outgoing push data towards the same dst. at regular intervals	TCP
7	DNS_Spike	Domain spike (>ten requests) and connection bursts from a host within a 60 second period	DNS
8	DNS_Isol	DNS request made without corresponding request by another application e.g., HTTP	DNS
9	UA_Blank	Empty User-Agent strings in HTTP header or malformedUA string	Web
10	Cert_Ano	Anomaly in TLS, certificate -subscriber and issuer organisation fields empty	Web
11	Mal_IPD	Direct access to destinationURL using IP address instead of domain name	Web
12	HTTP_Call	Callback HTTP traffic with only one isolated request to the domain	Web
13	DNS_Ent	Domain names with high entropy	DNS
14	DNS_TTL	Short Time-To-Leave, (TTL) in domain response (e.g., <=150secs)	DNS
15	DNS_Empty	Outbound DNS request, with no named query or encoded message	DNS
16	DNS_ NX	NX (non-existent, domains) responses in DNS reply	DNS
17	TCP_Retra	The number of reconnects between IP src/dst pair	TCP

2.2.4 Significant Challenges in Anomaly Detection

The task of accurately detecting threats is difficult. Current network security strategies, according to Akinrolabu et al., are primarily focused on detecting threats in individual network incidents, which has proven ineffective in detecting complex multi-stage attacks [47]. Furthermore, the people in charge of identifying threats are bombarded with a large number of false warnings from the current tools. As a result, advanced tools to detect advanced anomalies are desperately needed.

People related challenges in the threat detection.

- The notion of normalcy, it is one of the most critical steps in creating a network anomaly detection solution. The issue of "how to construct a precise definition of normality?" has prompted several researchers to develop various solutions over time. This is the most significant problem in anomaly detection, and it has yet to be completely solved. Many of the works discussed in this survey tried to achieve this goal [4].
- Attacks such as social engineering are out of the IDSs purview.
- Lack of human resources with correct skill set.

Operational challenges in the threat detection.

- Lack of adequate network and application logs - Users are unable to exchange log files because of data security concerns or a fear of overexposure to a company's assets [47].
- Adaptability - Any time new anomalies are added, or old ones are modified to circumvent existing detection solutions, anomalies change. As a result, IDSs must be revised on a regular basis to keep up with these improvements, which is not a simple job [4].
- Complexity - When researchers attempt to address all of the above problems, the complexity of the systems grows as new methods and approaches are added and mixed. Furthermore, the sophistication of today's network architectures leads to the persistence of this problem in terms of data collection and pre-processing [4].
- Real-time malware identification problems [38].

Technological challenges in the threat detection.

- To detect malware, there is a lack of automation, including the use of big data processing, cloud-related technology, and effective machine learning techniques [47].

- To prevent detection by IDS, evasion methods such as fragmentation, flooding, obfuscation, and encryption are used [16]. Furthermore, both Signature and Anomaly based IDSs have a difficult time detecting attacks disguised by evasion techniques.
- False alarm rates - Another issue is reducing false alarms as much as possible, but it is still impossible to fully eliminate them and create a 100% accurate IDS. That is always a challenge [4].
- Standard data sets - Only a few publicly accessible intrusion data sets with sufficient information about attacks are available; however, none of them is a standard assessment data set for anomaly detection. The lack of reliable public standard data sets that can accurately model network environments remains a challenge [4].
- Feature extraction is a challenging job since no one knows which approach would work the best [63].
- In creating an intrusion detector or a false alarm filter, choosing the right machine learning algorithm is crucial [63].

2.3 Zeek Network Monitoring System

Vern Paxson who is an American computer scientist designed and implemented the initial version of Bro (Now Zeek) two decades ago. Zeek is an open-source network traffic analyser that passively detects anomalies. It is primarily a security monitor that monitors all traffic on a link for signs of unusual activity. Zeek, on the other hand, supports a wide range of traffic analysis practices outside of the security context, such as performance evaluations and troubleshooting [65].

Network-based analytics platform Zeek Network Security Monitor (formerly Bro) converts network traffic into events and logs. Zeek provides users with a bird's-eye view of their network service using both signature-based and anomaly-based monitoring [37]. Hence, it is a popular NIDS among network security professionals [66].

Zeek differs from other network intrusion detection systems in that it focuses primarily on network monitoring activities. Zeek IDS are used to identify unique threats in the network and trigger warnings by documenting detailed network actions, unlike rules-based systems that are programmed to spot an exception [51]. Zeek has separate protocol analysis and policy script components, leaving the IDS approach open. It supports signature-based detection [13].

Zeek comes with a powerful set of characteristics [65].

Deployment

- Runs on standard UNIX-style systems on commodity hardware (including Linux, FreeBSD, and MacOS).
- Using a network tap or monitoring port, completely passive traffic analysis is possible.
- The libpcap interface is a standard interface for collecting packets.
- Analysis in real time and offline.
- Help for large-scale deployments in clusters.
- Operating all isolated and cluster setups with a unified management system.
- The software is free and open-source under the BSD license.

Analysis

- Detailed activity recording for offline research and forensics.
- Port-independent analysis of application-layer protocols.
- Support for many application-layer protocols (including DNS, FTP, HTTP, IRC, SMTP, SSH, SSL).
- File content shared over application-layer protocols is analysed, including MD5/SHA1 fingerprinting.
- Comprehensive IPv6 support.

- Detecting and analysing tunnels (including Ayiya, Teredo, GTPv1). Zeek decrypts the tunnels and then analyses their material as if there were no tunnel at all.
- During protocol analysis, extensive sanity tests are performed.
- Support for IDS-style pattern matching.

Scripting Language

- Turing-complete language for expression arbitrary analysis tasks.
- Event-based programming model.
- Domain-specific data types such as IP addresses (transparently handling both IPv4 and IPv6), port numbers, and timers.
- Extensive support for tracking and managing network state over time.

Interfacing

- Default output to well-structured ASCII logs.
- Alternative backends for Elasticsearch and DataSeries. Further database interfaces in preparation.
- Real-time integration of external input into analyses. Live database input in preparation.
- External C library for exchanging Zeek events with external programs. Comes with Perl, Python, and Ruby bindings.
- Ability to trigger arbitrary external processes from within the scripting language.

2.3.1 Zeek Architecture

Zeek is a platform for analysing network traffic that is both versatile and efficient. An event engine parses the traffic and produces a high-level event stream from incoming packets using protocol analysers [67]. The Zeek platform is made up of two sections. The first component is an event engine, which filters unnecessary data from captured network traffic before translating it into events. The interpreter, which runs scripts, is the second part. These scripts can specify acts to be performed in the event of certain situations or perform more complex event analysis [68] .

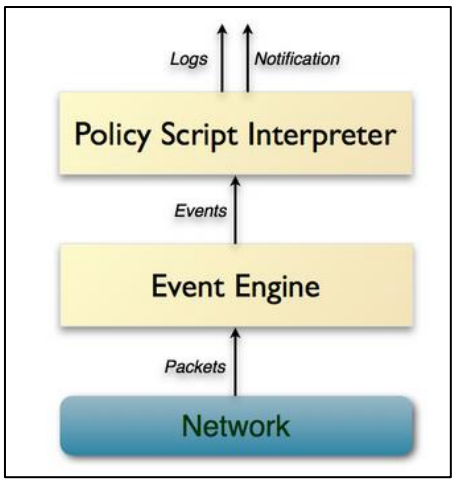


Figure 21: Zeek's Internal Architecture

As mentioned earlier the architecture of Zeek comprise with several components (As shown in Figure 21). In life, there are two main components. The event engine (or core) of the system divides the incoming packet stream into higher-level events. These events reflect network activity in a policy-neutral way, that is, they describe what has been observed but do not clarify why or why it is essential. On the wire, every HTTP request, for instance, generates an HTTP request event with the IP addresses and ports involved, the URI being requested, and the HTTP version in use. However, there is no

additional details included in the case, such as whether the URI corresponds to a known malware site. [65].

By executing a series of event handlers written in Zeek's custom scripting language, the script interpreter, Zeek's second main component, derives such semantics. These scripts may define a site's security policy, or what steps should be taken when the monitor detects various types of behaviour. They can obtain any desired properties and statistics from the input traffic in general. Zeek's vocabulary includes a variety of domain-specific styles and support functions, as well as the ability for scripts to maintain state over time, allowing them to monitor and compare how what they see evolves across connection and host boundaries. Zeek scripts can produce real-time warnings and run arbitrary external programs on demand, for example, to initiate an active response to an attack [65].

2.3.2 Zeek Log Files

For different Internet protocols, the Zeek system produces a number of log files. Since each log file has its own set of data, it is best to go over each one separately. An analysis of the log files is needed to better understand the significance of the IDS warning, which takes time and effort [69].

All HTTP sessions with their requested URIs, key headers, MIME forms, and server responses are included in the logs, as are DNS requests with replies, SSL certificates, SMTP session key content, and much more. All of this data is stored by Zeek in tab-separated log files that can be analysed by third-party tools. Users can, however, choose from a variety of alternative output formats and backends to directly interact with external databases, for example [65].

Listed below (Table 27) are few of the log files generated by Zeek [65]. Log files are generated and stored under $PREFIX/logs/current location.

Table 27: Logs Related to Network Protocols

Log File Name	Description
conn.log	TCP/UDP/ICMP connections
dce_rpc.log	Distributed Computing Environment/RPC
dhcp.log	DHCP leases
dnp3.log	DNP3 requests and replies
dns.log	DNS activity
ftp.log	FTP activity
http.log	HTTP requests and replies
irc.log	IRC commands and responses
kerberos.log	Kerberos
modbus.log	Modbus commands and responses
modbus_register_change.log	Tracks changes to Modbus holding registers
mysql.log	MySQL
ntlm.log	NT LAN Manager (NTLM)
ntp.log	Network Time Protocol
radius.log	RADIUS authentication attempts
rdp.log	RDP
rfb.log	Remote Framebuffer (RFB)
sip.log	SIP
smb_cmd.log	SMB commands
smb_files.log	SMB files
smb_mapping.log	SMB trees
smtp.log	SMTP transactions
snmp.log	SNMP messages
socks.log	SOCKS proxy requests
ssh.log	SSH connections
ssl.log	SSL/TLS handshake info
syslog.log	Syslog messages
tunnel.log	Tunneling protocol events

Log related to threat detection (Table 28).

Table 28: Logs Related to Threat Detection

Log File	Description
intel.log	Intelligence data matches
notice.log	Zeek notices
notice_alarm.log	The alarm stream
signatures.log	Signature matches
traceroute.log	Traceroute detection

Network observation related Zeek logs are shown in the following table (Table 29).

Table 29: Logs Related to Network Observation

Log File	Description
known_certs.log	SSL certificates
known_hosts.log	Hosts that have completed TCP handshakes
known_modbus.log	Modbus masters and slaves
known_services.log	Services running on hosts
software.log	Software being used on the network

Additional log files

Weird.log

This log tracks irregular or unusual behaviour that may imply malformed connections or traffic that does not follow a specific protocol.

Notice.log

Zeek identifies a single action as potentially interesting, unusual, or evil.

2.4 Summary

In chapter 2, we have performed an extensive literature survey related to the subject matter. We have discussed and explained how IDS technologies have evolved from the 1980s to modern-day IDSs. The types of IDSs and corresponding detection methods were discussed extensively throughout chapter 2. We believe that the information collated in Chapter 2 laid a firm foundation for us to proceed to Chapter 3. In chapter 3, we will dig dipper into the NIDS subject area and introduce a practical approach to improve the performance of a NIDS.

3. METHODOLOGY

Machine learning is predominantly useful for circumstances where it is challenging to develop code to solve a given problem. ML requires a sample of data or information on past knowledge to improve the performance of the output. Machine learning techniques can increase the accuracy, efficiency and remove the possibilities of human error. The rapid development in AI sector have lifted the application of ML in different sectors. Moreover, the ML technologies are resourceful in the Cybersecurity field. Research and development communities, academic researchers and corporates have invested millions of dollars to incorporate ML technologies to Cybersecurity. In our project we will be performing active network traffic analysis using the Zeek NSM and passive ML analysis using Weka application.

We have discussed the challenges and obstacles of threat detection and the practical implications of machine learning in threat detection in chapter one and two. This chapter focuses on designing a framework that actively and passively detects threats. In terms of machine learning we will analyse the network traffic and packet capture files (.pcap) using selected supervised learning methods, Decision tree and Decision table. We have designed and implement our system in a virtual environment. This environment is comprised of three actors (See Figure 22); Attacker, Victim machine, and NIDS.

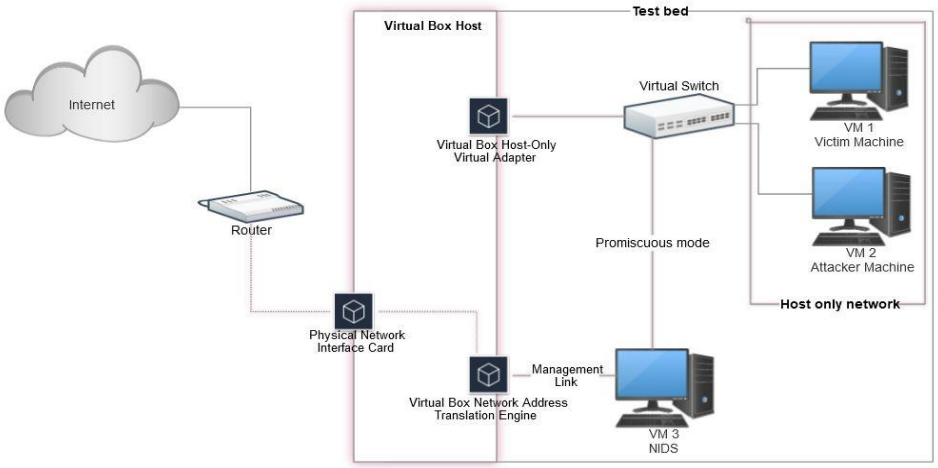


Figure 22: Test-bed

Moreover, we are expecting to integrate the Elasticsearch, Filebeat and Kibana (we call this integration as EFK stack) into our framework. EFK will assist us in collecting and processing network traffic data and provide tools to further analyse and visualise data.

3.1 Overview of Proposed Approach

The purpose of our research is to enhance the threat detection capabilities of a NIDS. We have come up with a stepwise process to achieve our objective (Figure 23). These steps are focussing on improving the threat detection capabilities using different type of technologies. The proposed NIDS solution require the existing knowledge of attacks and the behaviours native to such attacks. The correct identification with minimum false positives proves the effectiveness of NIDS and its threat detection capabilities.

The knowledge of existing attacks can be feed into the system by using data sets, predominantly by using .pcap files. Unfortunately, due to the lack of publicly published data, this is a challenging task. The notorious KDD'99 data set is used by researchers for some time, and it did not include novel attacks and abnormal behaviours. Hence, it is considered as one of the outdated data sets. We will be using a sample data set from UNSW-NB15 data set for our experiment.

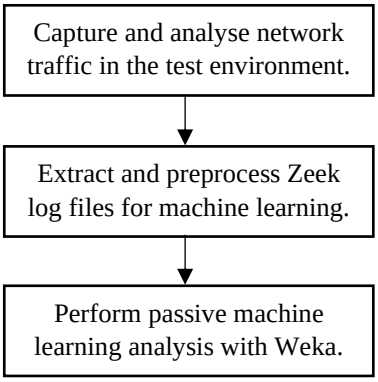


Figure 23: Overview of Proposed Approach

3.1.1 Capture and Analyse Network Traffic in the Test Environment

This section is focussed on exploring the Zeek's built-it threat detection capabilities. Moreover, we are expecting to assess Zeek's existing features/scripts that could be used to detect threats without using Machine learning algorithms.

Additionally, we are integrating the EFK stack with the intention of streamlining the threat detection and visualisation capabilities in our NIDS. The installation and the configuration of the Zeek and EFK stack is convoluted. Thus, this section will cover the Zeek NSM and EFK stack installation and configuration. We will walk you through the entire installation and configuration process while providing explanations in the upcoming sections.

3.1.2 Extract and Pre-process Zeek log Files for Machine Learning

In this section we will use the Zeek application mainly for log file generation, next the collated log files will be processed for ML analysis. Weka requires each input data set to be formatted in a .arff file format. In this section we will explore Attribute-Relation File Format (ARFF) in detail and perform the log file pre-processing procedure.

3.1.3 Perform Passive Machine Learning Analysis with Weka

Machine learning techniques are extensively used for IDSs, to identify the continuous changes in network attacks. The aim of this objective is to optimise the threat detection using Decision table and Decision tree classifiers. Our analysis will be performed on past packet capture files. Hence, we call it as a passive analysis.

3.2 Tools and Environments

Zeek application plays a major role in the proposed NIDS solution. Zeek is an open-source network traffic analyser. It can be used as a security monitor that inspects all network traffic against malicious activities [70]. It uses pcap files to capture IP packets and transfer them to an event engine that will accept or reject them. We will also use several other applications to improve the Zeek's threat detection capabilities:

3.2.1 Elasticsearch, Logstash and Kibana (ELK)

The Elastic Stack is also known as the ELK Stack is used to ingest data from many sources and in many different formats giving the end-user a way to visualise and analyse the data instantly. EFK stack is comprised of three components: (E)lasticsearch, (L)ogstash & (K)ibana. ELK stack is developed, managed, and maintained by an American-Dutch company named Elastic BV. ELK stack allows the users to create filters, analyse network log file, store log files, and visualise information [71] [72].

In our case we will not be using Logstash. Instead, we are going to use Filebeat. Going forward we will be calling the integration of Elasticsearch, Filebeat and Kibana as EFK stack.

3.2.1.1 Elasticsearch, Filebeat and Kibana (EFK)

Elasticsearch

This application is a free data analysis and search platform. It is an open-source suite based on Lucene. Elasticsearch supports real-time search and analysis functions. It can be used for reading and writing data, feature filtering. Analysis. Elasticsearch has the

characteristics of real-time search, stability, reliability, fast and easy to install and use [71].

Filebeat

Filebeat is the lightweight shipper, which we are going to use to ship logs from Zeek NSM to Elasticsearch. Filebeat is used for forwarding and centralising log files. Using Filebeat users can send thousands of logs from servers and Virtual Machines allowing for centralisation of the data. The Filebeat system can monitor, collect the log files, and transmit them for indexing. [73], In our case indexing will be done by Elasticsearch.

Kibana

Kibana is a data visualisation tool based on browser pages using Apache open-source protocol. This application is capable of handling logs visually and graphically and present the information through dashboards [71].

3.2.2 Weka

The term Weka stands for Waikato Environment for Knowledge Analysis. This application is developed at the University of Waikato, New Zealand. The Weka is an open-source, GUI based modern platform for applied machine learning. In fact, this is one of the best platforms for beginners getting started and practicing applied machine learning.

Weka entail with numerous machine learning algorithms that can be used to data mining and obtain significant information. Weka is written in Java language and

consist of a graphical user interface. It contains 49 information pre-preparing tools, 76 classification algorithms, 15 trait evaluators and 10 quest algorithms for highlight choice. It has three graphical user interfaces: “The Explorer”, “The Experimenter” and “The Knowledge Flow”. [74].

3.2.3 Network-based Data Sets

The efficacy of NIDS is measured by their ability to detect attacks, which necessitates a large data collection that includes both normal and abnormal behaviour. [75]. The data analysis algorithms of machine learning require continuous improvements. Well trained data sets facilitate corresponding algorithms to operate efficiently and effectively. As the name suggests, a data set is a collection of data records/instances, which shares common characteristics. Machine learning models require training data sets to perform effectively. It also requires testing data sets/validation data sets to make sure that the model is understanding training data correctly.

To train and evaluate anomaly-based network intrusion detection systems, labelled data sets are needed. Labeled data sets are needed for training supervised methods and testing both supervised and unsupervised intrusion detection techniques. Furthermore, in a labeled data set, the assessment parameters are focused on things like defining normal behaviours, the number of observed attacks, and the number of false alarms. [76]. We will be focussing on network traffic data sets from here onwards.

There are a few different ways to collect network traffic. The packet-based format or the flow-based format are most widely used. Mirroring ports on network devices is commonly used to capture network traffic at the packet level. The payload information is included in packet-based data. Flow-based data is more aggregated and usually includes only network link metadata. Flow-based network data is a compressed format that primarily includes meta data about network connections. Flow-based data combines all packets within a time window that share certain properties into a single

flow and typically does not have any payload. It is possible to convert packet-based data to flow-based data (but not vice versa) with tools like nfdump[2] or YAF[3] [76].

3.2.3.1 UNSW-NB15 Data Set

We have selected the UNSW-NB15 data set for our experiment. The UNSW-NB15 data set was created by Nour Moustafa and Jill Slay [70] using an IXIA PerfectStorm tool in the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS) to generate a hybrid of the realistic modern normal activities and the synthetic contemporary attack behaviours from network traffic [77]. Ring et al. [76] conducted a comprehensive literature survey on network-based intrusion detection and discuss the attributes of 34 data sets based on 15 properties. They have identified the UNSW-NB15 data set as a suitable data set for general NIDS evaluations procedures. Moreover, they have assessed the suitability of the UNSW-NB15 data set via 15 properties.

Authors have used Argus[4] and Zeek NSM tools to create reliable features from the pcap files. In addition, twelve algorithms are written in the C# programming language to analyse the connection packet flows in detail. A ground truth table containing all simulated attack forms is used to mark the data collection. This table was created using an IXIA report developed during the simulation. The UNSW-NB15 data set's main characteristics are a combination of actual modern normal behaviours and synthetical attack activities [77]. Properties of UNSW-NB15 data set are shown in Table 30. Following summary is take from the paper published by Ring et al [76].

[2] https://github.com/phaag/nfdump
[3] https://tools.netsa.cert.org/yaf/
[4] http://qosient.com/argus/index.shtml

Table 30: Properties of UNSW-NB15 Data Set

Property		Details	UNSW-NB15 Characteristics
General information	Year of traffic creation	Since network traffic is subject to concept drift and new attack scenarios appear daily, the age of an intrusion detection data set plays an important role. This property describes the year of creation.	2015
	Publicly available	Intrusion detection data sets should be publicly available to serve as a basis for comparing different intrusion detection methods. Furthermore, the quality of data sets can only be checked by third parties if they are publicly available.	Yes
	Normal user behaviour	This property indicates the availability of normal user behaviour within the data set.	Yes
	Attack traffic	IDS data sets should include various attack scenarios. This property indicates the presence of malicious network traffic within the data set.	Yes
Nature of the Data	Metadata	Metadata provide additional information about the network structure, IP addresses, attack scenarios and so on.	Yes
	Format	Network intrusion detection data sets appear in different formats. Other types of data sets may contain, e.g., packet-based traces with additional attributes such as host-based log files.	Packet-based network traffic, Other
	Anonymity	This property indicates if data is anonymised, and which attributes are affected.	None
Data Volume	Count	The property count describes a data set's size as either the number of contained packets/flows/points or the physical size in Gigabyte (GB).	100 GB
	Duration	The property duration provides the recording time of each data set.	31 hours[5]
Recording Environment	Kind of Traffic	This describes the origins of network traffic. Emulated means that real network traffic was captured within a test-bed or emulated network environment.	Emulated
	Type of Network	This property describes the underlying network environment in which the respective data set was created.	Small network
	Complete Network	This property indicates if the data set contains the complete network traffic	Yes

[5] The simulation period was 16 hours on Jan 22, 2015 and 15 hours on Feb 17, 2015 [70].

		from a network environment with several hosts, router and so on.	
Evaluation	Predefined splits	This property provides the information if a data set comes along with predefined subsets for training and evaluation.	Yes
	Balanced	Often, machine learning, and data mining methods are used for anomaly-based intrusion detection. In the training phase of such methods (e.g., decision tree classifiers), data sets should be balanced with respect to their class labels. Consequently, data sets should contain the same number of data points from each class (normal and attack). Real-world network traffic, however, is not balanced and contains more normal user behaviour than attack traffic. This property indicates if data sets are balanced with respect to their class labels. Imbalanced data sets should be balanced by appropriate preprocessing before data mining algorithms are used.	No
	Labeled	This property denotes if data sets are labeled or not. This is an important property for machine learning threat detection solutions.	Yes

Backdoors, Denial of Service (DoS), vulnerabilities, Fuzzers, Generic, Analysis, Reconnaissance, Shellcode, Worms are all covered by the UNSW-NB15 data collection. Table 31 provides a detailed description of records distribution of UNSW-NB15 data set [70].

Table 31: UNSW-NB15 Record Distribution

Type	Number of Records	Description
Normal	2,218,761	Natural transaction data
Fuzzers	24,246	Attempting to cause a program or network suspended by feeding it the randomly generated data.
Analysis	2,677	It contains different attacks of port scan, spam, and html files penetrations.
Backdoors	2,329	A technique in which a system security mechanism is bypassed stealthily to access a computer or its data.
DoS	16,353	A malicious attempt to make a server or a network resource unavailable to users, usually by temporarily interrupting or suspending the service of a host connected to the Internet

Exploits	44,525	The attacker knows of a security problem within an operating system or a piece of software and leverages that knowledge by exploiting the vulnerability
Generic	215,481	A technique works against all block ciphers (with a given block and key size), without consideration about the structure of the block-cipher.
Reconnaissance	13,987	Contains all Strikes that can simulate attacks that gather information.
Shellcode	1,511	A small piece of code used as the payload in the exploitation of software vulnerability.
Worms	174	Attacker replicates itself to spread to other computers. Often, it uses a computer network to spread itself, relying on security failures on the target computer to access it.

Features are the distinct characteristics of a phenomenon that you want to explore in machine learning. The features you provide determine how strong machine learning classifiers are. As a consequence, one of the most critical tasks in machine learning is to come up with good features. Following that, we will go through the various types of features available in our UNSW-NB15 data collection.

The authors have introduced 49 features with the UNSW-NB15 data set. These features are divided into six groups based on their characteristics [70]. Moustafa et al. have describe the 6 categories as follows [78] :

- Flow features: This category includes the identifier attributes between hosts, such as client-to-serve or server-to-client.
- Basic features: This category involves the attributes that represent protocols connections.
- Content features: This group captures the attributes of TCP/IP; also, they contain some attributes of http services.
- Time features: This category encompasses the attributes of time, for example, arrival time between packets, start/end packet time and round-trip time of TCP protocol.
- Additional generated features: This group has further divided into two sub-groups.
    - General purpose features: Each feature of this group has its own purpose, to protect the service of protocols.

- Connection features: Connection features are solely created to provide defence during attempt to connection scenarios [70].

- Labelled features: This group represents the label of each record.

UNSW-NB15 data set is available in multiple forms[6]. In fact, data is available in .pcap files, .csv format, as Zeek log files and as Argus files. We will be using the data available in .csv format. The total number of records is two million and 540,044. The authors [78] have stored them in the four CSV files, namely, UNSW-NB15_1.csv, UNSW-NB15_2.csv, UNSW-NB15_3.csv and UNSW-NB15_4.csv[7].

Below table (Table 32) presents more details of UNSW-NB15 features.

Table 32: Features of UNSW-NB15

No.	Group	Feature Name	Description
1	Flow Features	srcip	Source IP address
2		sport	Source port number
3		dstip	Destination IP address
4		dsport	Destination port number
5		proto	Transaction protocol
6	Basic Features	state	The state and its dependent protocol. e.g., ACC, CLO
7		dur	Record total duration
8		sbytes	Source to destination bytes
9		dbytes	Destination to source bytes
10		sttl	Source to destination time to live
11		dttl	Destination to source time to live
12		sloss	Source packets retransmitted or dropped
13		dloss	Destination packets retransmitted or dropped
14		service	Protocols such as http, ftp, ssh, dns
15		sload	Source bits per second
16		dload	Destination bits per second
17		spkts	Source to destination packet count
18		dpkts	Destination to source packet count
19	Content Features	swin	Source TCP window advertisement
20		dwin	Destination TCP window advertisement
21		stcpb	Source TCP sequence number
22		dtcpb	Destination TCP sequence number

[6] URL : https://cloudstor.aarnet.edu.au/plus/index.php/s/2DhnLGDdEECo4ys

[7] URL : https://cloudstor.aarnet.edu.au/plus/index.php/s/2DhnLGDdEECo4ys?path=%2FUNSW-NB15%20-%20CSV%20Files

23		smeansz	Mean of the flow packet size transmitted by the src
24		dmeansz	Mean of the flow packet size transmitted by the dst
25		trans_depth	The depth into the connection of http request/response transaction
26		res_bdy_len	The content size of the data transferred from the server's http service.
27	Time Features	sjit	Source jitter (mSec)
28		djit	Destination jitter (mSec)
29		stime	Record start time
30		ltime	record last time
31		sintpkt	Source inter-packet arrival time (mSec)
32		dintpkt	Destination inter-packet arrival time (mSec)
33		tcprtt	The sum of 'synack' and 'ackdat' of the TCP.
34		synack	The time between the SYN and the SYN_ACK packets of the TCP.
35		ackdat	The time between the SYN_ACK and the ACK packets of the TCP.
36	Additional generated features General purpose features	is_sm_ips_ports	If source (1) equals to destination (3) IP addresses and port numbers (2)(4) are equal, this variable takes value 1 else 0
37		ct_state_ttl	No. for each state (6) according to specific range of values for source/destination time to live (10) (11).
38		ct_flw_http_mthd	No. of flows that has methods such as Get and Post in http service.
39		is_ftp_login	If the ftp session is accessed by user and password, then 1 else 0.
40		ct_ftp_cmd	No of flows that has a command in ftp session.
41	Additional generated features Connection features	ct_srv_src	No. of connections that contain the same service (14) and source address (1) in 100 connections according to the last time (26).
42		ct_srv_dst	No. of connections that contain the same service (14) and destination address (3) in 100 connections according to the last time (26).
43		ct_dst_ltm	No. of connections of the same destination address (3) in 100 connections according to the last time (26).
44		ct_src_ltm	No. of connections of the same source address (1) in 100 connections according to the last time (26).
45		ct_src_dport_ltm	No of connections of the same source address (1) and the destination port (4) in 100 connections according to the last time (26).
46		ct_dst_sport_ltm	No of connections of the same destination address (3) and the source port (2) in 100 connections according to the last time (26).
47		ct_dst_src_ltm	No of connections of the same source (1) and the destination (3) address in in 100 connections according to the last time (26).

3.2.4 Test-bed Specifications

The test-bed was implemented on a system with the following configuration:

- Core - Intel Core i7-1065G7 CPU @ 1.30GHz 1.50 GHz
- Disk - 500GB HDD
- RAM - 16 GB
- Operating system - Windows 10
- Virtualisation Product – Oracle Virtualbox 6.1.16

Virtual Machines (VMs)

Specifications of the VMs are listed in Table 33.

Table 33: Test-bed Virtual Machines

	Network Security Monitor (NIDS)	**Attacker Machine**	**Vitim Machines (#2)**
Machine Name	NSM_VM01	Attacker_VM02	windows7_p1 & Metasploitable_project1
Operating System	Ubuntu 20.04.2.0 desktop	Kali linux-2021-1	Windows 7 Professional - Service Pack 1 & Metasploitable2.6.24-16. server (Ubuntu)
Processor (#)	2	1	1
Disk	50GB	15GB	15GB & 8GB
Ram	4GB	2GB	2GB & 1GB
IP Address	192.168.10.7	192.168.10.10	192.168.10.2 &192.168.10.6
Network Adaptor	-NAT (Management) -Host-only Adapter (Sniffing Port)	Host-only Adapter	Host-only Adapter

Applications in Use

Table 34 shows the information on the applications we used in our project.

Table 34: Applications in Use

Application Name	Version	Purpose
Zeek Network Security Monitor	2.3.0	Zeek will be used to analyse the network behaviours and generate logfiles for further analysis
Elasticsearch	7.11.2	Elasticsearch performs indexing and storing of incoming file feed from Filebeat
Filebeat	7.11.2	Filebeat collects log files from Zeek and feed them into Elasticsearch
Kibana	7.11.2	Kibana Visualise the network behaviour to the user
Weka	3.8.5	Application we used to apply machine learning classifier to analyse captured packets
Java – Openjdk version	1.8.0_282	Required as a pre-requisite for Weka and EFL

It is important to have all elastic related applications (EFK) in the same version.

3.3 The Proposed NIDS Solution

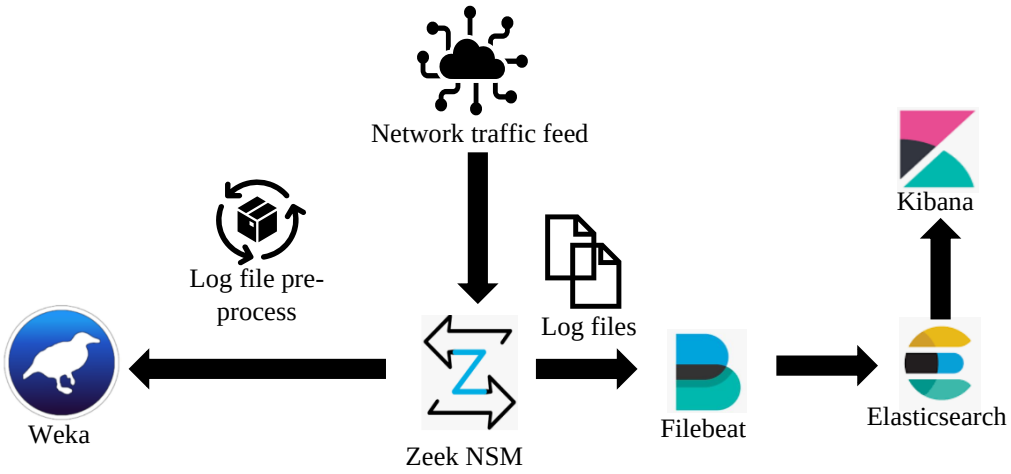


Figure 24: High-level Architecture to Improve Threat Detection Capabilities

The NIDS solution has three main components. The high-level architecture of the proposed NIDS solution is depicted in Figure 24. Zeek NSM application, the EFK and Weka application. Zeek NSM is responsible for capturing network traffic data and feed them into EFK stack. But that is not the only responsibility of Zeek application. As we mentioned in the previous chapter, Zeek is highly configurable through its scripting language and it provides detailed log information. Hence, we expect to study the

existing detection capability of the Zeek application in detail by focussing on selected functionalities in chapter 5.

Secondly, we integrate the EFK stack into the proposed architecture. EFK stack will further analyse the log information more effectively and monitor the network system. Moreover, the Filebeat collect and filter Zeek logs and feed them into Elasticsearch. Then, Elasticsearch stores data passed in by Filebeat and feed its input to the Kibana dashboard.

Finally, we focus on integrating machine learning techniques to our framework using the Weka application. In the next chapter, we will systematically implement the proposed solution and assess the threat detection capabilities using several use cases. Furthermore, we expect to cover the application installation, configuration, and deployment process in more detail in the coming chapter.

3.4 Summary

This concludes the methodology that we are planning to use for the proposed NIDS solution. In this chapter, we have discussed the applications and technologies we are going to use in the implementation chapter. In fact, we have presented our three-step approach in 3.1 Overview of Proposed Approach We discussed the specifics of EFK stack, Weka application and the importance of network-based data sets. In the latter part of the methodology chapter, we discussed the characteristics of the UNSW-NB15 data set. In the next chapter, we are going to implement the proposed NIDS in a virtual environment.

4. IMPLEMENTATION

The implementation section is divided into three sections.

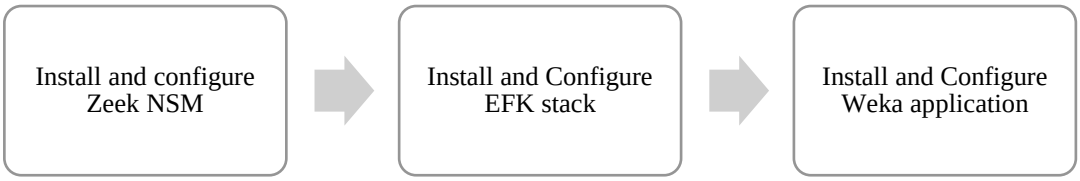


Figure 25: Implementation Plan

We have explained the installation and configuration procedures in detail in the APPENDICES section. The following sections will take you through how our NSM framework captures network traffic and perform further analysis. The implementation plan is shown in Figure 25. Subsection 4.1 to 4.3 will present the basic functionality of our framework. This includes the following:

- Performing online and offline packet capturing and analysis using Zeek application and EFK stack.
- Logfile Pre-processing procedure
- Perform machine learning analysis using selected machine learning classifiers.

Moreover, from 4.1 to 4.3 demonstrate a scenario, where the attacking machine (Attacker_VM02) perform a Nmap[8] scan on a victim machine (Metasploitable_project1) and demonstrate how the NSM respond to the situation. Then, in subsection 4.2 we will take you through the log file pre-processing procedure and in 4.3 the pre-processed logs fed into Weka for machine learning analysis using decision table and j48 classifiers.

From subsection 4.4 we expand our scope a bit further and perform the analysis using the UNSW-NB15 data set. UNSW-NB15 is a large data set. Due to storage limitations,

[8] Network Mapper or the Nmap is a free and open-source application, which is used for network discovery and security auditing activities.

we cannot incorporate the entire data set for our testing. Hence, we will be selecting a subset of UNSW-NB15 data to perform the analysis.

4.1 Capture and Analyse Network Traffic in the Test Environment

In this section, we are going to perform a Nmap scan from the Attacker_VM02 machine to Metasploitable_project1 machine. Now that we have configured all the components in our NSM solution, Zeek should be able to capture the network behaviour and feed the information into the EFK stack.

As the first step, we will be performing a Nmap scan on a victim machine, Table 35 explains the tasks and the expected outcome of this use case. associated screenshots are depicted in Figure 26. Moreover, Figure 26 shows the attacking machine performing the Nmap scan on 192.168.10.6 (Victim Machine) and the NSM generate logs in /opt/zeek/logs/current location. Lastly, Kibana shows subsequent network behaviour on its dashboard (Highlighted in red color box).

Table 35: Use Case - Online Network Traffic Analysis

Use case 1: Online Network Traffic Analysis		
Task	**Expected Outcome**	**Actual outcome**
An Attacker performing a Nmap scan on the victim machine	• Attacker should be able to perform a successful Nmap scan from 192.168.10.10 to 192.168.10.6 • Zeek should generate log files. • EFK stack should capture the scenario and feed the information to Kibana	As expected,

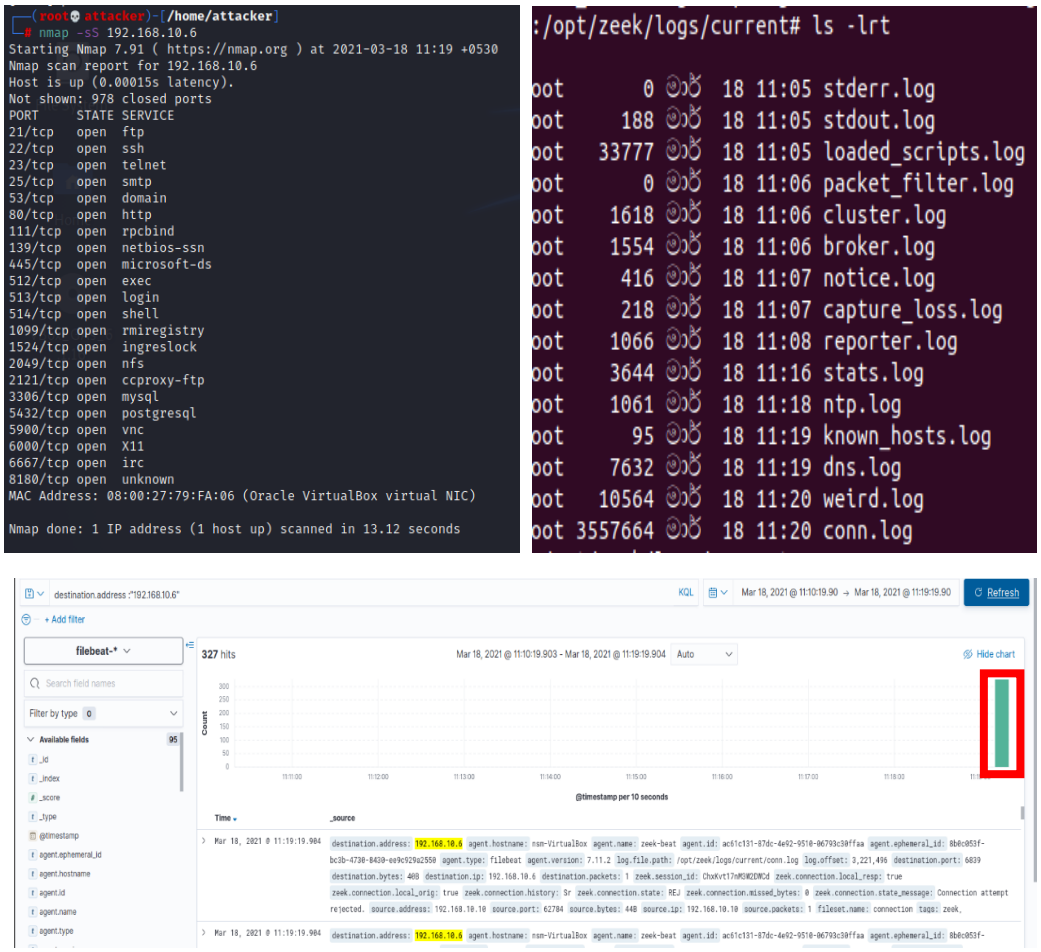


Figure 26: Online Network Traffic Analysis

Table 36, explains the process to perform an offline network traffic analysis using Zeek utilities. We have used smallFlows.pcap for this exercise (This pcap is readily available to use in https://s3.amazonaws.com/tcpreplay-pcap-files/smallFlows.pcap URL). Figure 27 shows the screen captures of the user running the Zeek utility on smallFlows.pcap files and Zeek generating corresponding log files in the NSM.

Table 36: Use Case - Offline Network Traffic Analysis

Use case 2: Offline Network Traffic Analysis		
Task	**Expected Outcome**	**Actual outcome**
Generate Zeek log files by reading from an offline packet capture file	For this task we require a sample .pcap file to perform the testing.[9] • Perform Zeek analysis on smallFlows.pcap file. • Zeek should generate corresponding log files	As expected,

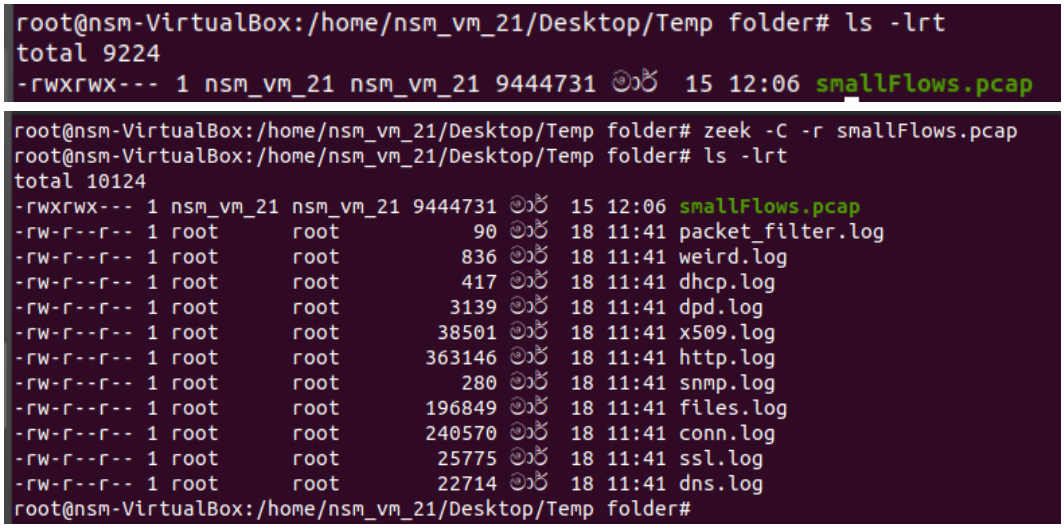


Figure 27: Zeek Offline Network Traffic Log Generation

Table 37, illuminates the process to perform Offline Network Traffic Analysis by Replaying the smallFlows.pcap file using the tcpreplay utility. As you can see in Figure 28, we are performing the packet capture replay on the sniffing interface (enp0s8). Zeek NSM and Kibana dashboard (see the red color box) working as expected by capturing the network behaviour.

Table 37: Use Case - Offline Network Traffic Analysis by Replaying .pcap file

Use case 3: Offline Network Traffic Analysis by Replaying .pcap file		
Task	**Expected Outcome**	**Actual outcome**
Generate Zeek log files by replaying .pcap files	We will be using the same .pcap file, which we used in use case 2 for this test.	As expected,

[9] Available in: https://s3.amazonaws.com/tcpreplay-pcap-files/smallFlows.pcap

	• Replay an offline .pcap file on the sniffing interface. • Zeek should generate corresponding log files. • EFK stack should capture the scenario and feed the information to Kibana. The sniffing interface is set to enp0s8 network interface. *Command:* tcpreplay -t -v -i enp0s8 smallFlows.pcap	

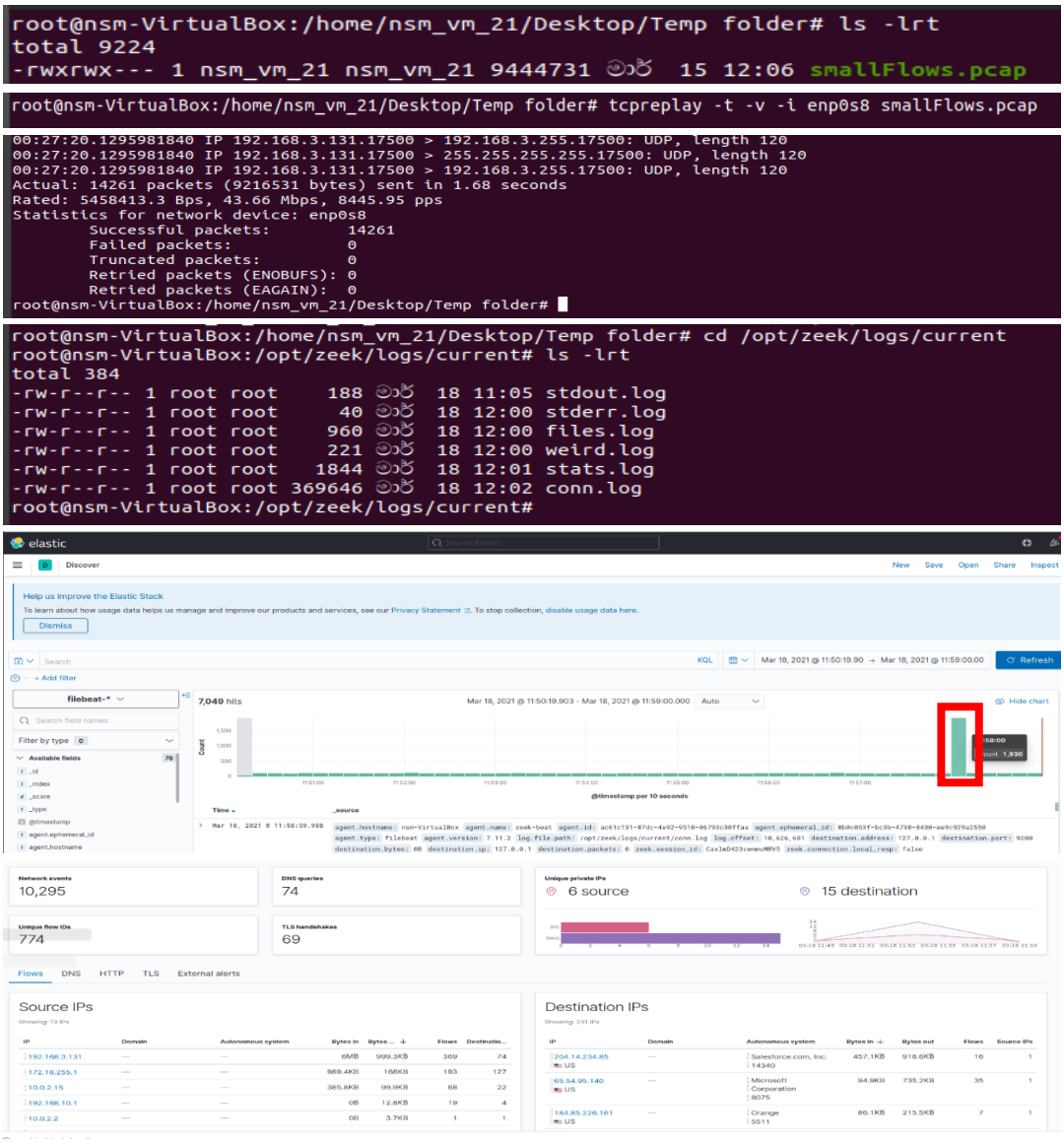


Figure 28: Offline Network Traffic Analysis by Replaying .pcap file

4.2 Extract and Pre-process Zeek log Files for Machine Learning

Now we have proved that our NSM is functional and capable of capturing network activities we can proceed to the second part of our framework. We will further investigate Zeek functionalities and log file analysis in subsection 4.4.

As we mentioned earlier in Chapter 2, false-positives and false-negatives detection have a significant impact to IDSs solutions. We have incorporated machine learning capabilities to detect such instances. The objective of this section is to perform the prerequisite tasks for machine learning analysis. In fact, we will take you through the file pre-processing procedure, training data set creation procedure, testing data set creation procedure and .arff file creation procedure.

4.2.1 Data Pre-Processing Procedure

Unfortunately, we cannot feed the Zeek output files directly to Weka. The input files of Weka should be in .arff (Attribute-Relation File Format) format. Weka also entertains the .json (JavaScript Object Notation) formatted files and .csv (Comma-separated values) formatted files. Since our NSM is producing its log files in .tsv (Tab-separated values) format we will be converting .tsv files to .arff files using Linux utilities (we have listed the commands in Appendix G: Data Pre-processing Process section) . The high-level diagram of Pre-processing procedure is shown in Figure 29.

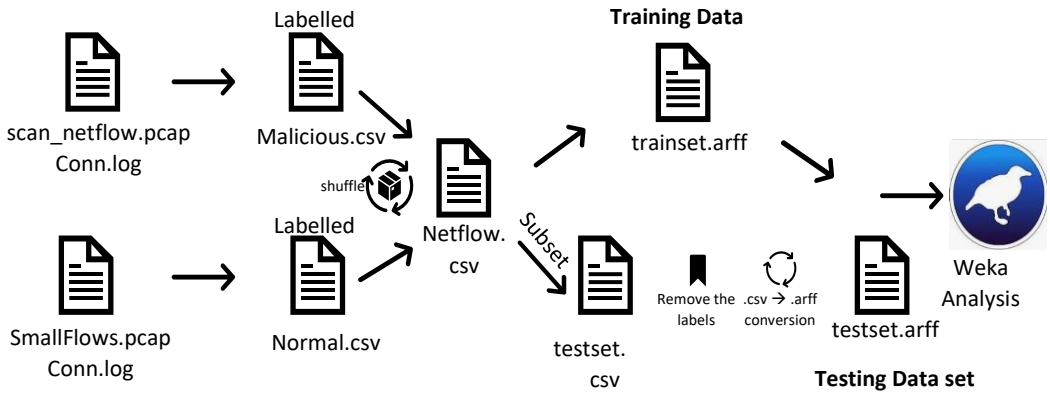


Figure 29: Data Pre-Processing Procedure

Data pre-processing procedure (Table 38) has few objectives:

- To create and label Malicious/Abnormal behaviour file (Figure 31,Figure 32).
    - Here, the Malicious content could be Nmap scans, DoS attacks, SQL Injections etc.
- To create and label a Normal behaviour file (Figure 33,Figure 34).
- To combine Malicious/Abnormal behaviour file and Normal behaviour file into one file (Netflow.csv)
- To generate a Training set and a Test data set from the Netflow.csv file.
- To remove labels from Test data set
- To generate trainset.arff file and testset.arff (Figure 35,Figure 36)

As shown in the above figure we are using two .pcap files to generate conn.log.

- scan_netflow.pcap: Contains the Malicious Behaviours (Nmap scan)
- smallFlows.pcap: Contains the Normal Network Traffic

Table 38: Use Case - Creating .arff Files Using Conn.logs

<table>
<tr><td colspan="3">Use case 4: Creating .arff files using Conn.logs</td></tr>
<tr><th>Task</th><th>Expected Outcome</th><th>Actual outcome</th></tr>
<tr><td colspan="3">Use Case 4.1: Create scan_netflow.pcap file.
We will be capturing a NMAP scan for this example.</td></tr>
<tr><td>• Launch packet capture on interface enp0s8 and save the output into a file named scan_netflow.pcap.

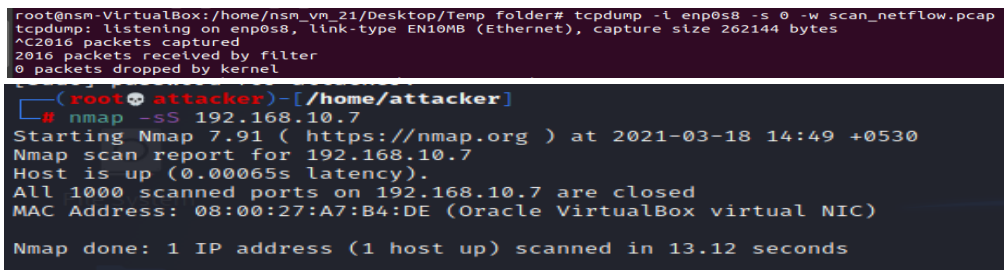
• Perform a Nmap scan from the attacker’s machine</td><td>• scan_netflow.pcap should get generated</td><td>As expected,</td></tr>
<tr><td colspan="3">Screen Capture


Figure 30: Use Case - Create scan_netflow.pcap file</td></tr>
</table>

Use Case 4.1.1: Generate Conn.log using Zeek utilities		
• Run Zeek command on scan_netflow.pcap file to generate corresponding log files	Corresponding log files should get generated	As expected,

Screen Capture

```
root@nsm-VirtualBox:/home/nsm_vm_21/Desktop/Temp folder# zeek -C -r scan_netflow.pcap
root@nsm-VirtualBox:/home/nsm_vm_21/Desktop/Temp folder# ls -lrt
total 9504
-rwxrwx--- 1 nsm_vm_21 nsm_vm_21 9444731 මාර්  15 12:06 smallFlows.pcap
-rwxrwx--- 1 nsm_vm_21 nsm_vm_21    3421 මාර්  15 13:46 runbook.txt
-rw-r--r-- 1 tcpdump   tcpdump    147891 මාර්  18 14:50 scan_netflow.pcap
-rw-r--r-- 1 root      root          254 මාර්  18 14:52 packet_filter.log
-rw-r--r-- 1 root      root          919 මාර්  18 14:52 dns.log
-rw-r--r-- 1 root      root       119570 මාර්  18 14:52 conn.log
```

Figure 31: Use Case - Generate Conn.log using Zeek utilities

Use Case 4.2: Generate Malicious.csv file
• Use linux utlities to remove . and " " using linux utilies such as sed, awk, cat as Weka do not accept such character in the input files. • Lable the records as "1" for identification purposes

Screen Capture

```
 1 1295981542708292,1921683131,7214213102,55950,80,tcp,0058485,1
 2 1295981543461968,1921683131,2074614838,55955,80,tcp,0028620,1
 3 1295981543337241,1921683131,65551737,55954,80,tcp,1776718,1
 4 1295981546609581,1921683131,20882236129,58264,80,tcp,0125448,1
 5 1295981546736952,1921683131,20882236129,58265,80,tcp,0169843,1
 6 1295981549760088,1921683131,7214213105,57721,443,tcp,0001363,1
 7 1295981549832444,1921683131,20882236129,58272,80,tcp,0166319,1
 8 1295981543841133,1921683131,655495140,55963,80,tcp,9427326,1
 9 1295981545127681,1921683131,655495142,55973,80,tcp,8140921,1
10 1295981543652521,1921683131,206108207139,55960,80,tcp,17762163,1
11 1295981561406421,1921683131,742175010,57038,80,tcp,0081750,1
12 1295981562220872,1921683131,7214213102,52201,443,tcp,0067879,1
13 1295981562221263,1921683131,7214213102,52203,443,tcp,0068419,1
```

Figure 32: Malicious.csv file

Use Case 4.3: Generate Conn.log for smallFlows.pcap file. (This pcap file taken from the internet)		
• Run Zeek command on smallFlows.pcap file to generate Zeek log files	Corresponding logfiles should get generated	As expected,

Screen Capture

```
root@nsm-VirtualBox:/home/nsm_vm_21/Desktop/Temp folder# zeek -C -r smallFlows.pcap
root@nsm-VirtualBox:/home/nsm_vm_21/Desktop/Temp folder# ls -lrt *.log
-rw-r--r-- 1 root root 236096 මාර්  18 14:55 http.log
-rw-r--r-- 1 root root    673 මාර්  18 14:55 dhcp.log
-rw-r--r-- 1 root root  88942 මාර්  18 14:55 conn.log
-rw-r--r-- 1 root root  21385 මාර්  18 14:55 x509.log
-rw-r--r-- 1 root root    725 මාර්  18 14:55 weird.log
-rw-r--r-- 1 root root  16010 මාර්  18 14:55 ssl.log
-rw-r--r-- 1 root root    502 මාර්  18 14:55 snmp.log
-rw-r--r-- 1 root root    254 මාර්  18 14:55 packet_filter.log
-rw-r--r-- 1 root root  93829 මාර්  18 14:55 files.log
-rw-r--r-- 1 root root   2110 මාර්  18 14:55 dpd.log
-rw-r--r-- 1 root root  10946 මාර්  18 14:55 dns.log
```

Figure 33: Use Case - Generate Conn.log for smallFlows.pcap file

Use Case 4.3: Generate Normal.csv file
• Use linux utlities to remove . and " " using linux utilies such as sed, awk, cat as Weka do not accept such character in the input files. • Label the records as "0" for identification purposes

Screen Capture

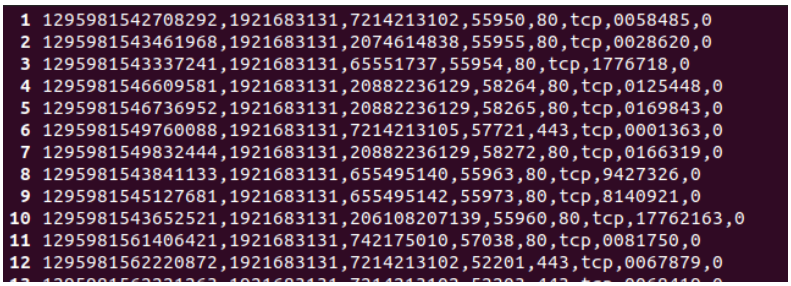


Figure 34:Normal.csv file

Use Case 4.4: Generate Netflow.csv file

In this instance, we shuffle and combine the Normal.csv file and Malicious.csv file and create a single file called Netflow.csv using the following command.

Command:
cat normal.csv malicious.csv > Netflow_temp.csv
shuf Netflow_temp.csv > Netflow.csv

Use Case 4.5: Generate trainset.csv file

This data set is used to train the classifier. It contain both normal and malicious records.

Use Case 4.6: Generate testset.csv file

Take a subset from Netflow.csv file and generate testset.csv file. These records should not be available in trainset.csv. Malicious records are labelled for classifier optimisation purposes.

Use Case 4.7: Generate trainset.arff file

Add Weka related attributes and generate the .arff file.

Screen Capture

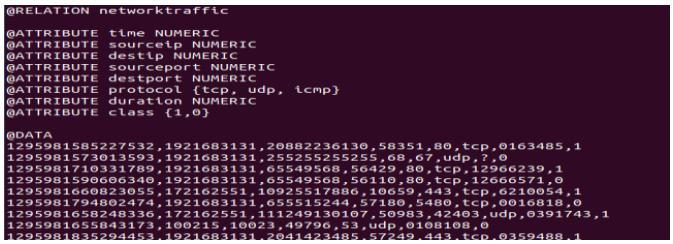


Figure 35: trainset.arff file format

Use Case 4.6: Generate testset.arff file

Add Weka related attributes and generate the .arff file. Remove the flag/labels column: “0”,”1” as the Weka should predict the label.in the ML testing process

Screen Capture

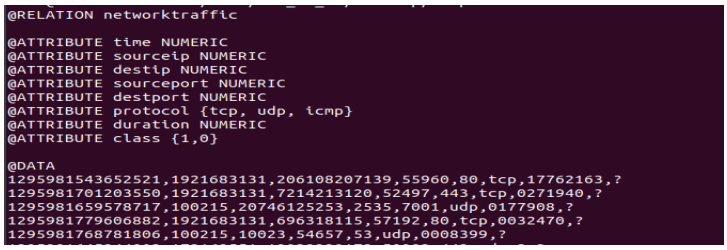


Figure 36: testset.arff file format

4.2.2 Feed Data Files to Weka Application

We have created the .arff file in the previous stage. Let us feed that information to the Weka application.

Open Weka GUI → Click on Pre-process tab → Click Open file → Select a .arff file → click Ok. We have successfully loaded the .arff file to Weka (Figure 37).

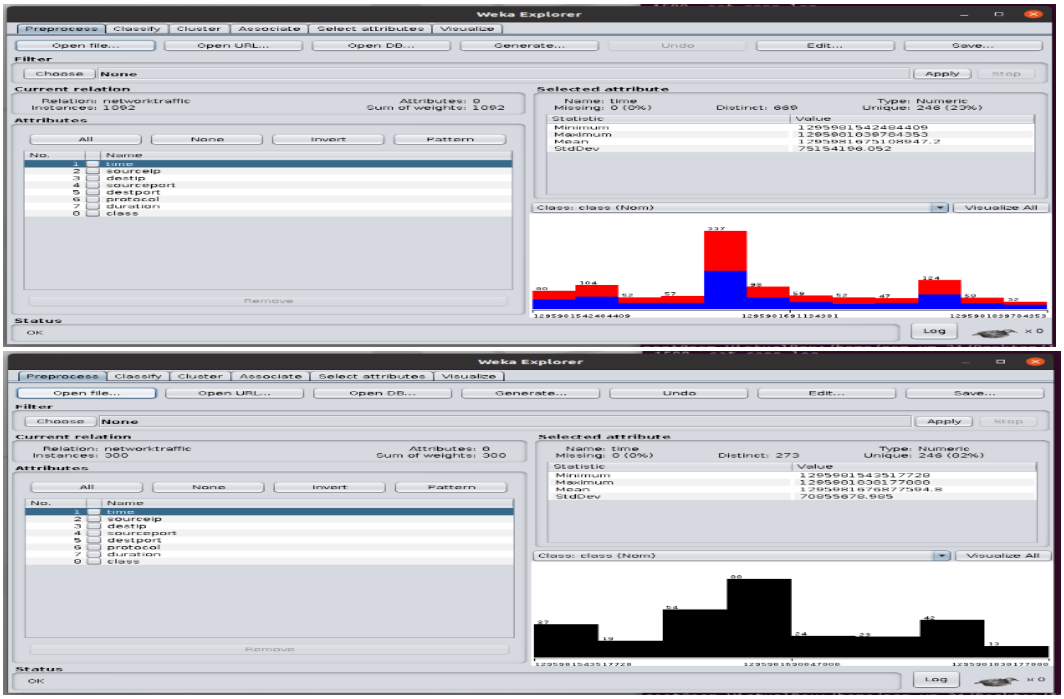


Figure 37: Feed .arff files to Weka Application (Training Data and Testing Data)

4.3 Perform Passive Machine Learning Analysis with Weka

In this section, we are going to feed the data sets to the Weka application and train a machine learning model to detect Nmap scans. We will use the decision table and decision tree(J48) supervised algorithms for this test. These algorithms are available in the Weka application. Refer to Appendix F: Weka Installation section for the application installation procedure.

As the first step, insert the training data set into the Weka application. Weka will use Training data set (trainset.arff) to train the machine learning classifier on Nmap scan

behaviours. Upon the data insertion, Weka will update its GUI as per to the .arff file settings (Figure 38).

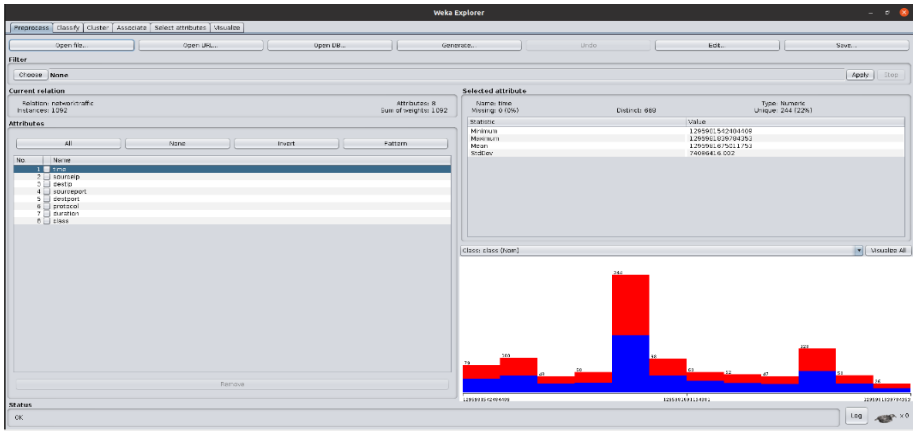


Figure 38: Weka Explorer GUI

Second step is to train a decision table classifier using Weka. Following are the training results performed by Weka application (Figure 39).

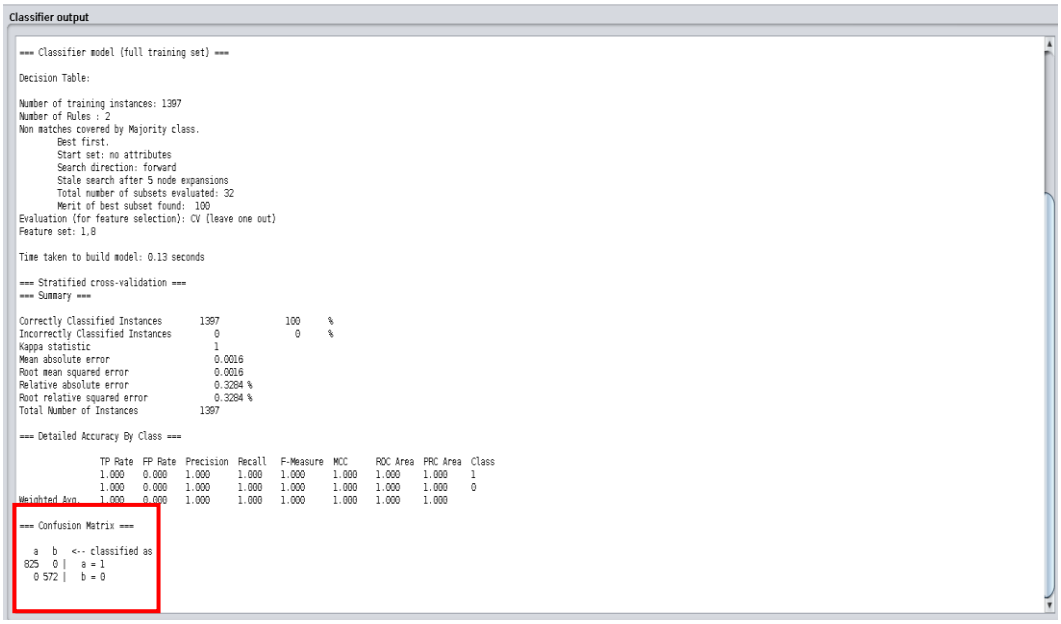


Figure 39: Results of the Decision Table Algorithm

One of the important aspects of the results is the Confusion Matrix. We have discussed on confusion matrixes in chapter 2. Following confusion matrix (Table 39) of the

Decision table algorithm shows the classifier has 100% accuracy in predicting labels after being trained.

Table 39: Confusion Matrix - Decision Table Algorithm

825	**0**
0	**572**

Next, let us train the same data set with a different classifier. We used Decision Tree(J48) algorithm for the second instance. Results are shown in Figure 40.

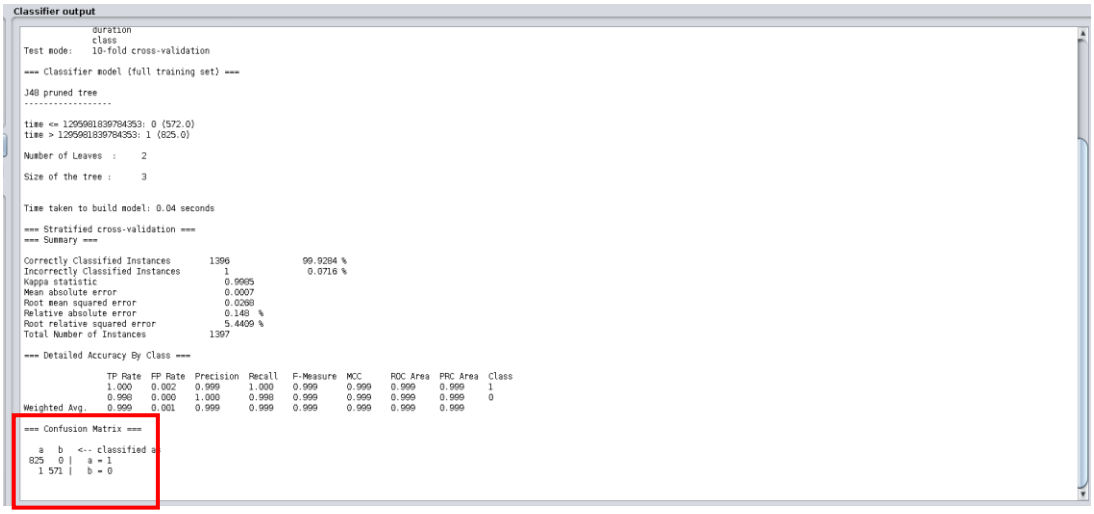


Figure 40: Results of the J48 Algorithm

The results in the confusion matrix shows the classifier did not have 100% accuracy like in the previous scenario. As per the confusion matrix results (Table 40), the classifier has labelled one malicious data packet as normal. This is an example of a false-negative detection.

Table 40: Confusion Matrix - J48 algorithm

825	**0**
1	**571**

Weka allows users to improve the classifiers accuracy by removing or adding features to the classifiers. As shown in (Figure 41), only decision node in the tree is time feature. As the data was collated in different time periods the decision tree have relied on the time feature to ascertain when the network events took place.

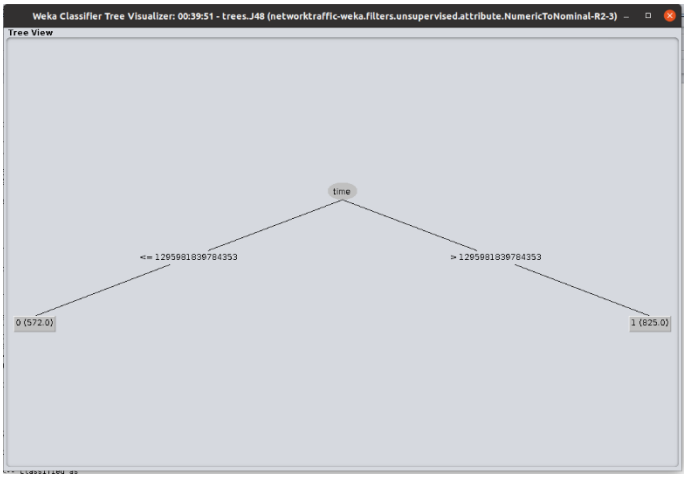


Figure 41: Decision tree Reliance on the Time Feature

Figure 42 shows the Weka results of decision tree classifier after removing the time feature. The confusion matrix results in Table 41 shows that the inaccuracy has increased after the modification.

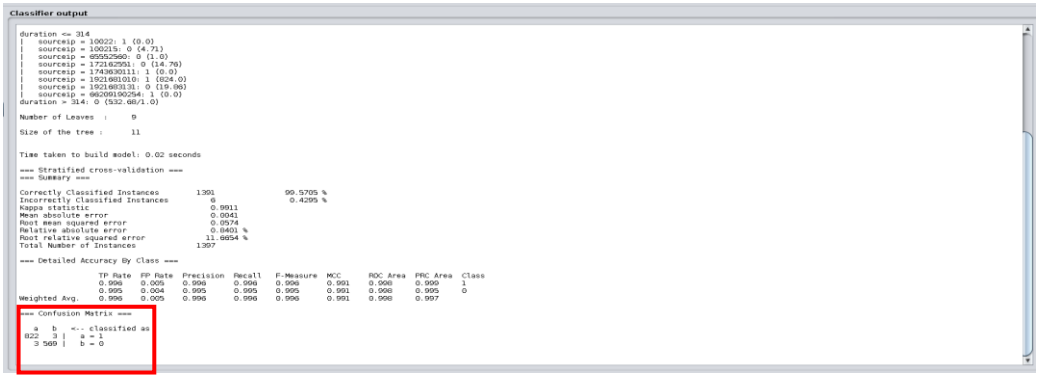


Figure 42: Results of the J48 Algorithm After the Re-configuration

Table 41: Confusion Matrix - J48 algorithm After Re-configuration

822	**3**
3	**569**

This is a good example to show the importance of choosing the right features and suitable classifier for ML analysis. Not all the features are necessary and required at every scenario. Hence, it is important to select your features carefully according to the requirement.

Since we had a good result on the Decision table algorithm (Table 39). Let us create a Machine learning model from the Decision table and use it to predict the records in testset.arff file. Note that the last column of testset.arff is currently set to empty by a question mark (Figure 43). The rationale behind this is for Decision table model to predict the record available in testset.arff file and output the results. Moreover, malicious traffic (NMAP instances) will be marked as "1" and benign traffic will be marked as "0".

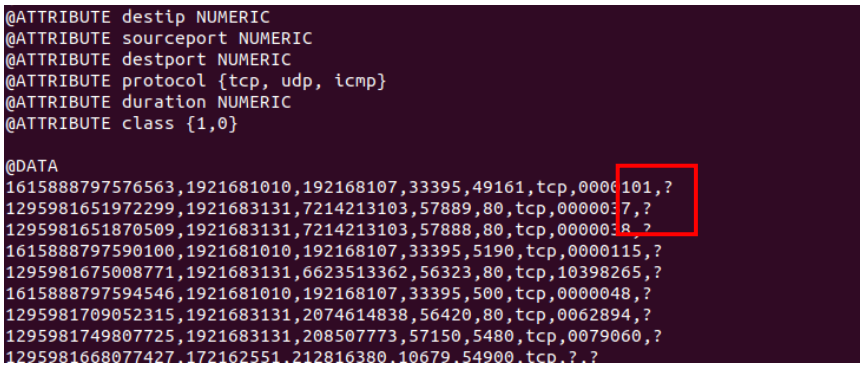


Figure 43: Unprocessed testset.arff

Following are the steps that need to perform to create the ML predictive model using the Decision Table algorithm and to test the testset.arff file to perform the predictions via Weka.

Load the trainset.arff file to Weka and go to the classify tab and train the data set using Decision Table algorithm (Choose → Weka → Classifiers → Rules → Decision Table). Next add the testset.arff to Weka application by selecting the "Supplied test

set" radio button on classify tab and then click on the set button to select the testset.arff file from the corresponding file location. In order to perform the predictive analysis, locate the trained model from the Result list and right-click on it to see options. Next select the "Re-evaluate model on current test set" option from the options list. This will perform the Weka predictive analysis on the testset.arff file.

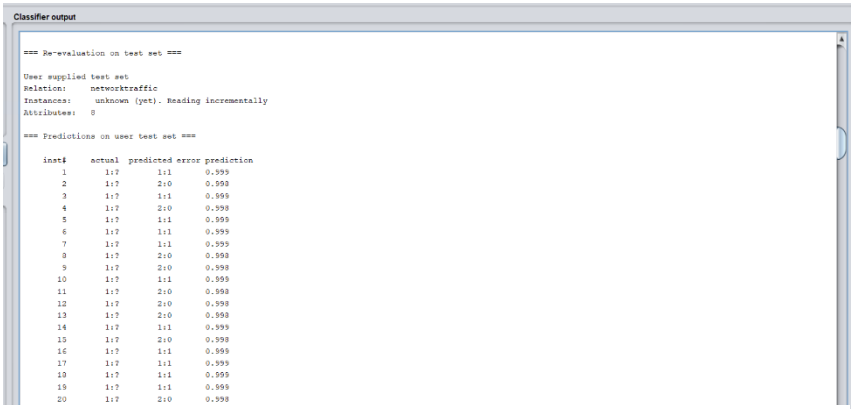


Figure 44: ML Model Predictions

In order to run the predictive model in the test set. We have loaded the trained Decision table algorithm and re-evaluate the model using testset.arff using Weka. Figure 44 depicts the output information of Weka, full list is provided in Appendix J. Overall the result shows 100% accuracy. We have cross-checked the output of the ML model with the test.csv (Figure 45) and construct the confusion matrix for the Decision Table Predictive model (Table 42).

First ten records of test.csv with the labels

D	E	F	G	H	I
48788	82	tcp	28	1	
56455	80	tcp	4020841	0	
48788	4242	tcp	368	1	
57248	8443	tcp	36234262	0	
48788	57797	tcp	181	1	
48788	9290	tcp	151	1	
48788	2251	tcp	3	1	
50983	32575	udp	6580849	0	
50983	23389	udp	100287	0	
48788	32770	tcp	160	1	

* labels are shown in the H column
* 1 denotes a malicious record & 0 denotes a benign record

First ten records of Decision Table Predictive Model Output

```
=== Predictions on user test set ===

 inst#     actual  predicted error prediction
     1        1:?        1:1       0.999
     2        1:?        2:0       0.998
     3        1:?        1:1       0.999
     4        1:?        2:0       0.998
     5        1:?        1:1       0.999
     6        1:?        1:1       0.999
     7        1:?        1:1       0.999
     8        1:?        2:0       0.998
     9        1:?        2:0       0.998
    10        1:?        1:1       0.999
```

*predicted labels are shown in predicted error column

test.csv records	Predictive Model Results	
1	1	Matched
0	0	Matched
1	1	Matched
0	0	Matched
1	1	Matched
1	1	Matched
1	1	Matched
0	0	Matched
0	0	Matched
1	1	Matched

Figure 45: First 10 instances of test.csv file and the output of predictive model

The confusion matrix is an effective tool because it tells you how classes were classified or misclassified graphically. As per Table 42, 190 of malicious traffic have been correctly identified by the model while 109 benign records are also correctly identified by the system.

Table 42: Confusion Matrix - Decision Table Predictive Model

190	**0**
0	**109**

Weka also produce graphs and diagrams which shows the association between the features and the data distribution of the classifiers (Figure 46).

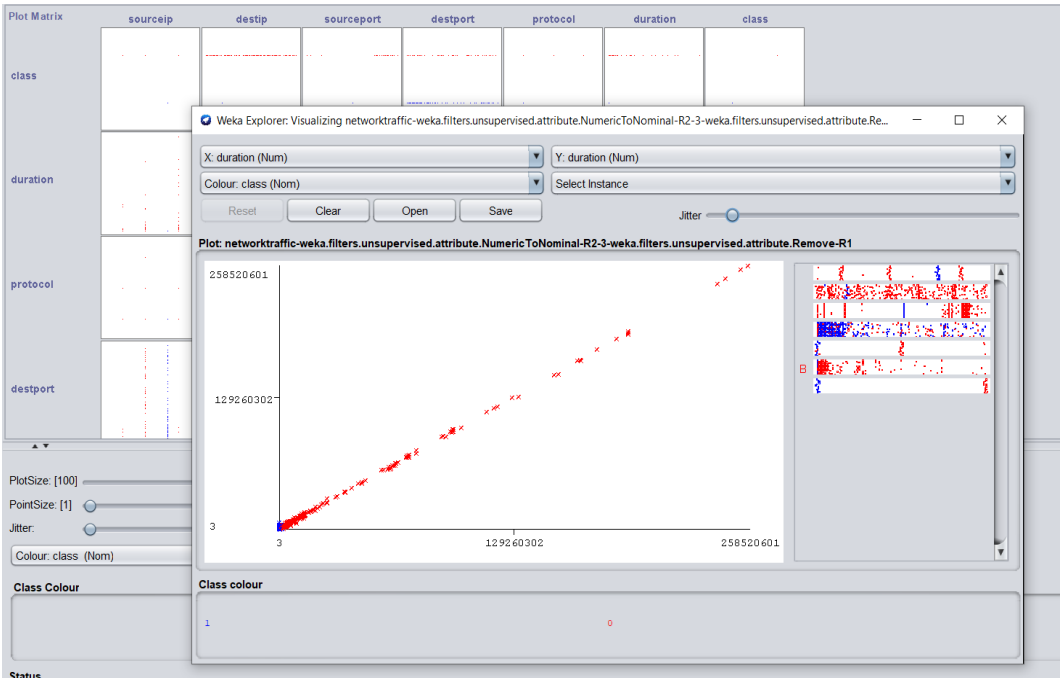


Figure 46: Visualising the Predicted Labels for the Test Data Set

We have now shown you that our framework can actively monitor network traffic and perform passive machine learning analysis using the Weka application. In the next

section, we are expecting to select features based on the literature review done on UNSW-NB15 data set and perform Weka analysis on selected network attacks.

4.4 Training ML Models Using a Sample Set from the UNSW-NB15 Data Set

Next, we will analyse a sample data set of UNSW-NB15 and perform a Weka analysis by training three Supervised machine learning models. Namely, we are expecting to train Random Forrest Algorithm, K-nearest Neighbor Algorithm (KNN) and Naïve Bayes using the Weka application. The KNN classifier is mentioned as IBK in the Weka application. The UNSW-NB15 data set provides network-traffic information on several attacks (Table 31). For our project, we are limiting to three attack types. In fact, we will be focussing on (i) Denial-of-service attacks (DoS), Network Reconnaissance instances and Worm attacks. The rationale behind the above selection is based on the following aspects. The definitions of these attacks are discussed in Table 1.

DoS Attacks:

- DoS attacks are one of the top variants in the incidents report in 2019 [79].
- DoS attacks on IoT devices tripled in the first half of 2019 [80]
- Number of DDoS attacks: Attacks will double to 15.4 million by 2023 globally [81]

Above statistics shows the importance of implementing defences against DoS attacks. Hence, we have selected this attack variant from the UNSW-NB15 data set.

Reconnaissance Attacks:

- Reconnaissance techniques entails gathering the information about the victim. Basically, it is the initial step of the Cyber Criminals before they launch attacks [82]
- Tools such as NMAP, trace, strobe, Nessus, WHOIS, port scanner (an advanced port scanner), nslookup, ping are commonly used tools for reconnaissance attempts [83]

Reconnaissance is the initial step of most Cyber-attacks as it is the requirements gathering phase of Cybercriminals to launch attacks. Thus, it is important to implement defences against reconnaissance attacks. The above facts led us the way to add this attack type to our sample.

Worm Attacks:

- In 2020 H1, worms were frequent attacks [84]
- Famous Stuxnet attack is also a worm attack
- WannaCry was a computer worm responsible for one of the most devastating cyberattacks in recent history [85]

Above facts highlight the importance of protecting IT systems from worm attacks.

Table 43 shows the number of records available in the UNSW-NB15 data set, which are related to DoS, Reconnaissance and Worm attacks. Going forward we will call them as classes.

Table 43: UNSW-NB15 No. Records on the Selected Attacks

Type	Number of Records	Description
DoS	16,353	In DoS attacks, the attacker makes the internet services unavailable by flooding the target server with phony network requests.
Reconnaissance	13,987	Reconnaissance is performed to gather information on the vulnerable areas of a target system
Worms	174	In Worm attacks the malicious content replicate itself to spread to other systems.

Feature selection is an important factor when training a machine learning model. Selecting inappropriate features will provide misguiding statistics on the data set. In this project, we will not be doing a feature selection. Instead, we will be referring to past literature and select the most suitable features for our three classes. The study conducted by Moustafa et al. [78] presents the subsets of features for different attack

types. Based on their findings the Table 44 list down the most frequently used features of DoS, Reconnaissance and Worm attacks.

Table 44: UNSW-NB15 Features Proposed by ARM Algorithm

Type	No. Features	Suggested Features	Description
DoS	11	state	The states and its dependent protocol e.g., CON
		dttl	Destination to source time to live
		sload	Source bits per second
		dload	Destination bits per second
		is_sm_ips_ports	If srcip = dstip and sport = dsport , assign 1 else 0
		ct_state_ttl	No. of each state according to values of sttl and dttl
		is_ftp_login	If the ftp session is accessed by user and password, then 1 else 0
		ct_ftp_cmd	No of flows that has a command in ftp session
		ct_srv_dst	No. of rows of the same service and dstip in 100 rows
		ct_src_ltm	No. of rows of the srcip in 100 rows
		ct_src_dport_ltm	No of rows of the same srcip and the dsport in 100 rows
Reconnaissance	11	sttl	Source to destination time to live
		service	Such as http, ftp, smtp, ssh, dns and ftp-data
		ct_state_ttl	No. of each state according to values of sttl and dttl
		ct_srv_src	No. of rows of the same service and srcip in 100 rows
		ct_srv_dst	No. of rows of the same service a dstip in 100 rows
		ct_dst_ltm	No. of rows of the same dstip in 100 rows
		ct_src_ltm	No. of rows of the srcip in 100 rows
		dbytes	Destination to source bytes
		dload	Destination bits per second
		spkts	Source to destination packet count.
		djit	Destination jitter
Worms	11	ct_srv_src	No. of rows of the same service and srcip in 100 rows
		ct_state_ttl	No. of each state according to values of sttl and dttl
		dbytes	Destination to source bytes
		dttl	Destination to source time to live
		sttl	Source to destination time to live
		ct_dst_sport_ltm	No of rows of the same dstip and the sport in 100 rows
		smeansz	Mean of the packet size transmitted by the srcip
		spkts	Source to destination packet count
		service	Such as http, ftp, smtp, ssh, dns and ftp-data
		proto	Protocol type, such as TCP, UDP
		dloss	Destination packets retransmitted or dropped

We will be considering the above features for our ML model training. Figure 47 shows the stepwise approach to test the selected sample data of UNSW-NB15. The process is further describing in Figure 48 in more detail. We will be using 80:20 splitting principle for the testing and training activities. It is said that the 80:20 ratio tends to provide accurate results [86].



Figure 47: Procedure to Test the Selected Sample Data

The following sections will perform ML analysis on each class. Moreover, subsequent sections will present the results using Weka. In section 5.1 Findings, we will be further analysing the results

The .csv file contains 30,514 records, after removing unnecessary data and features. It contains 22 features. Refer to Table 44 for feature distribution details. We have removed the following features due to incompleteness. Moreover, the below features only applicable to the DoS class.

is_ftp_login → If the ftp session is accessed by user and password then 1 else 0

ct_ftp_cmd → No of flows that has a command in ftp session.

This reduces the total number of features to 20. Also, we have replaced "-" with "?" in the last column. Weka only recognise "?" as an empty value. Finally, we have added corresponding table headers to the .csv file and upload the master.csv file to Weka application.

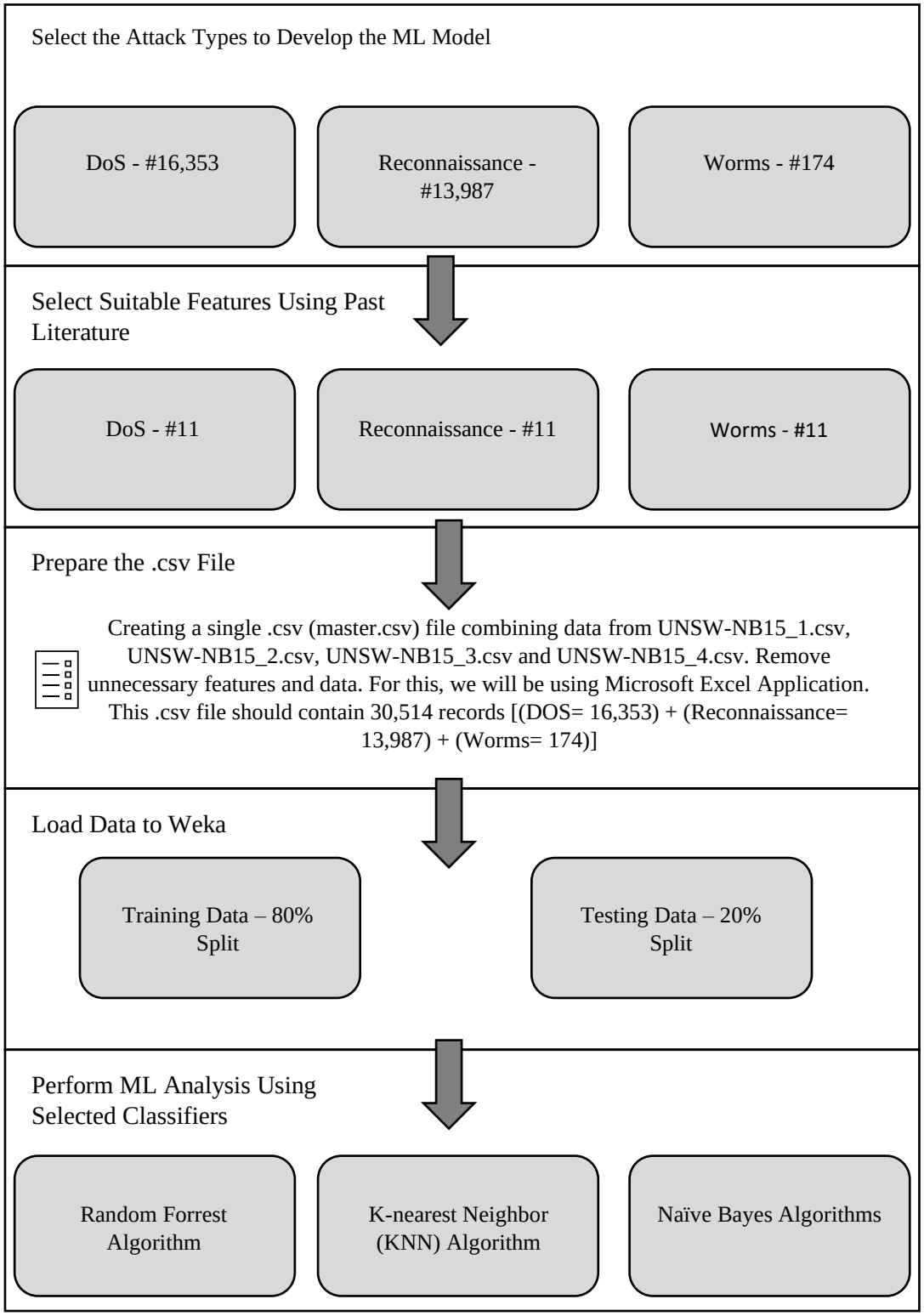


Figure 48: UNSW-NB15 Sample Data Set Testing Framework

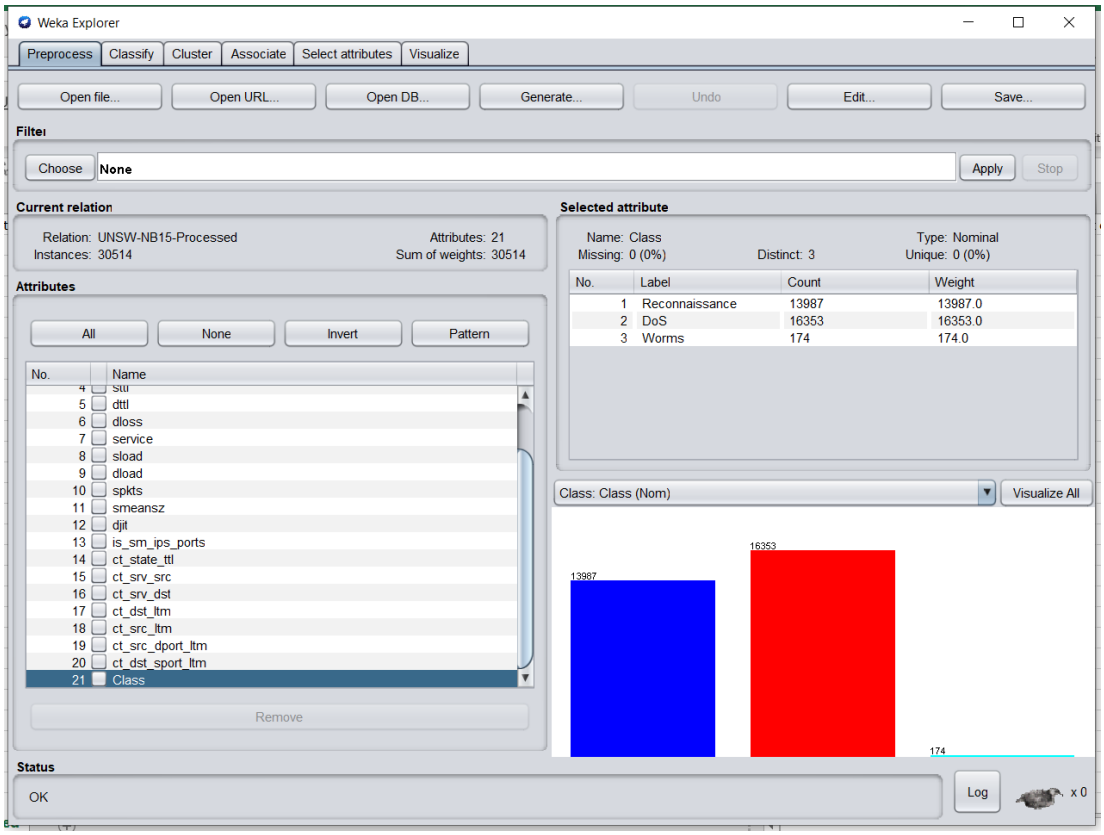


Figure 49: Upload the Master.csv file to Weka

Figure 49 portrays that, data set is not balanced. Worms attack class has the lowest records from all three classes. The Red column represent the DoS class, Blue column represent the Network Reconnaissance activities and the Light-blue colour bar represent the Worms attacks. Weka also generated the graphs, which shows the relationships between the attributes (Features) and the classes (Figure 50).

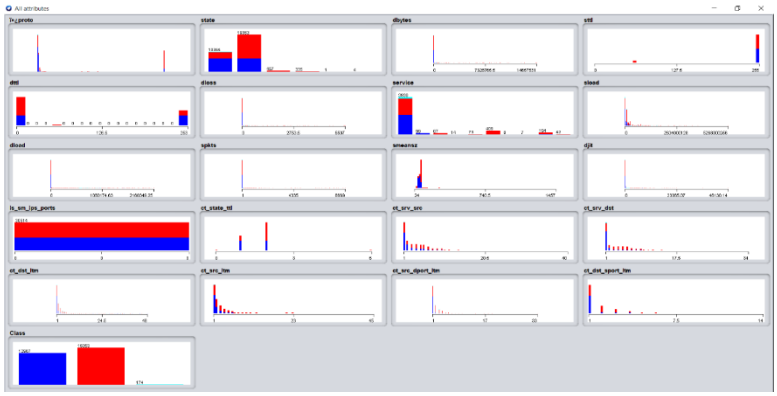


Figure 50: Relationship Between the Attributes (Features) and the Classes

4.4.1 Training the ML Model

We have now created our data set. Let us run the data set using the selected classifiers (#3). We will be analysing the results in the 5.1 Findings section. Refer to Appendix H: Weka – Classifier Outputs section for information on Weka outputs.

Results of the ML Analysis – Random Forrest Classifier

Procedure: Load the data set → Set the type of Class feature to Nominal → Go to Classify tab → Click on Choose → Go to trees category → Select RandomForrest → Select Percentage split (We have set the to 80%) → Make Sure, that you have selected (Nom) Class from the dropdown menu → Click on Start for model training.

The output screen of Weka is presented in Figure 51.

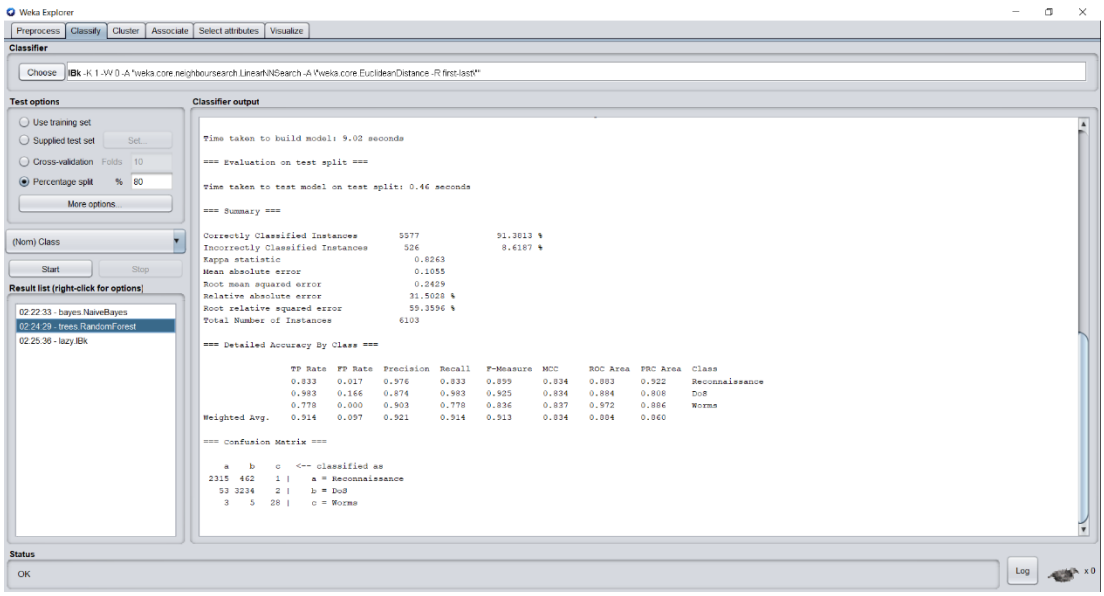


Figure 51: Result of the ML Analysis – Random Forrest Classifier

Results of the ML Analysis – KNN Classifier

Procedure: Load the data set → Set the type of Class feature to Nominal → Go to Classify tab → Click on Choose → Go to lazy category → Select IBk → Select Percentage split (We have set the to 80%) → Make Sure, that you have selected (Nom) Class from the dropdown menu → Click on Start for model training.

The output screen of Weka is presented in Figure 52

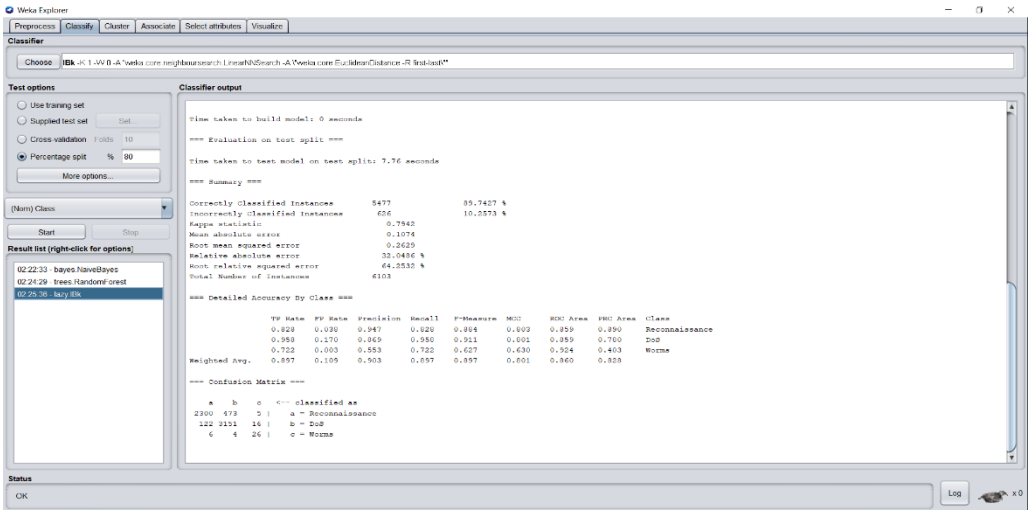


Figure 52: Result of the ML Analysis – KNN Classifier

Results of the ML Analysis – Naïve Bayes Classifier

Procedure: Load the data set → Set the type of Class feature to Nominal → Go to Classify tab → Click on Choose → Go to bayes category → Select Naïve Bayes → Select Percentage split (We have set the to 80%) → Make Sure, that you have selected (Nom) Class from the dropdown menu → Click on Start for model training. The output screen of Weka is presented in Figure 53.

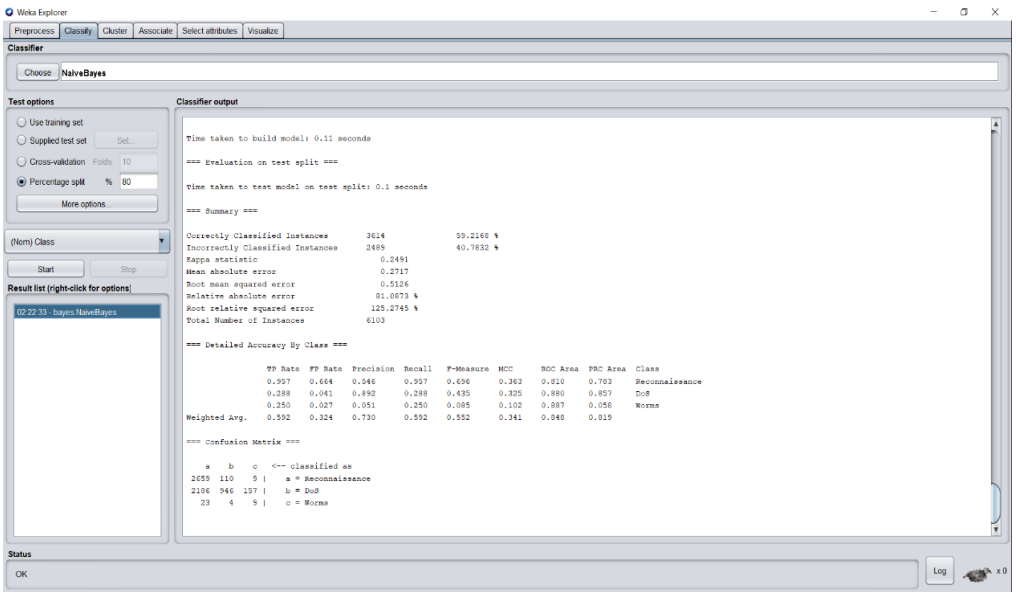


Figure 53: Result of the ML Analysis – Naïve Bayes Classifier

4.5 Summary

This concludes chapter 4. In chapter 4 we built and tested a prototype implementation of a NIDS. We have also tried to incorporate machine learning technologies into our framework and performed multiple experiments using a sample data set from UNSW-NB15 data set. In chapter 5 we will analyse our findings and we are expecting to discuss the challenges we faced during the implementation and the future plans of the authors.

5. DISCUSSION AND CONCLUSIONS

In this section we will take you through the following sections:

- Findings
- Challenges and limitations, we faced throughout the project.
- Future work related to the project.

5.1 Findings

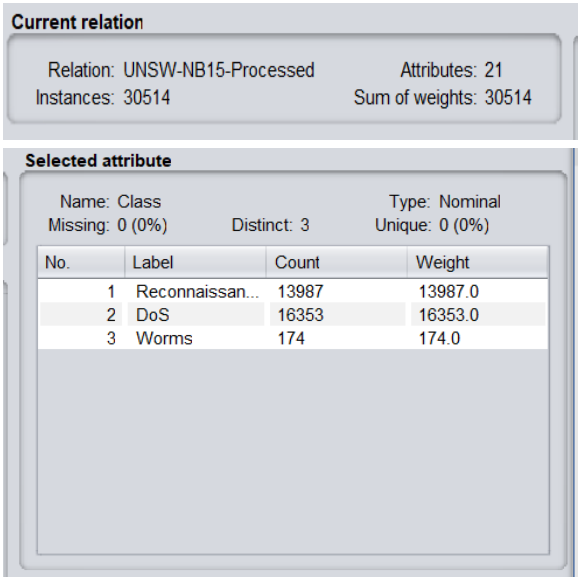


Figure 54: Weka Interface - Current Relation & Selected Attribute

In this section, we are expecting to take you through the findings we discovered by running tests in Weka. We have tested 30,514 records using 20 attributes/features. As depicted in Figure 54, our data set contains three well-known Cyber-attacks. Network Reconnaissance, DoS Attacks and Worm attacks. Before moving into the statistics let us go through the different measures, we will see in the Weka application.

- TP Rate represent the True-positive rate
- FP Rate represent the False-positive rate
- Precision describes the fraction of corresponding instances among rest of the instances.
- Sensitivity or the recall explains a fraction of corresponding instances that were obtained. Hence, Precision and Sensitivity are based on relevance.
- Combine measure for Sensitivity and Precision represented by F-Measure value.
- Receiver Operating Characteristics (ROC) area measurement represent the general performance of the model.

In the following tables, we have highlighted the best performing values in green colour. and low performing values in red colour.

5.1.1 Discussion on the Random Forrest Classifier Training

Table 45: Detailed Accuracy by Class - Random Forrest

Class	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area
Reconnaissance	0.833	0.017	0.976	0.833	0.899	0.834	0.883	0.922
DoS	0.983	0.166	0.874	0.983	0.925	0.834	0.884	0.808
Worms	0.778	0.000	0.903	0.778	0.836	0.837	0.972	0.886
Weighted Avg.	0.914	0.097	0.921	0.914	0.913	0.834	0.884	0.860

Table 46: Confusion Matrix - Random Forrest

Reconnaissance	DoS	Worms	
2315	462	1	**Reconnaissance**
53	3234	2	**DoS**
3	5	28	**Worms**

Random Forrest Classifier has performed well in the test. It has managed to come to the top in all 8 measures (Table 45). The Confusion matrix (Table 46) results are fairly good. In fact,

- The ideal Confusion Matrix should only have numbers in the yellow colour cells (Diagonal line).
- The classifier has labelled 2315 Reconnaissance instances correctly and 463 instances incorrectly.
- Random Forrest algorithm managed to identify 3234 DoS instances correctly. Unfortunately, it has labelled 53 DoS attacks as Reconnaissance and 2 DoS attacks as Worm attacks.
- Finally, the Random Forrest classifier manages to identify 28 Worm attacks correctly and it has missed 8 Worm instances in the training process.

5.1.2 Discussion on KNN Classifier Training

Table 47: Detailed Accuracy by Class - KNN

Class	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area
Reconnaissance	0.828	0.038	0.947	0.828	0.884	0.803	0.859	0.890
DoS	0.958	0.170	0.869	0.958	0.911	0.801	0.859	0.780
Worms	0.722	0.003	0.553	0.722	0.627	0.630	0.924	0.403
Weighted Avg.	0.897	0.109	0.903	0.897	0.897	0.801	0.860	0.828

Table 48: Confusion Matrix - KNN

Reconnaissance	DoS	Worms	
2300	473	5	**Reconnaissance**
122	3151	16	**DoS**
6	4	26	**Worms**

The KNN classifier manages to come to the middle in the standings. This classifier has performed averagely (Table 47). The Confusion Matrix of KNN has mixed results. In Table 48,

- The ideal Confusion Matrix should only have numbers in the yellow colour cells (Diagonal line).
- The classifier has correctly labelled 2300 records as Reconnaissance and 478 Reconnaissance instances incorrectly.

- KNN algorithm managed to identify 3151 DoS instances correctly. Unfortunately, it has labelled 122 DoS attacks as Reconnaissance and 16 DoS attacks as Worm attacks.
- Finally, this classifier manages to identify 26 Worm records correctly and it has missed 10 Worm attacks in the process of training.

5.1.3 Discussion on Naïve Bayes Classifier Training

Table 49: Detailed Accuracy by Class - Naïve Bayes

Class	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area
Reconnaissance	0.957	0.664	0.546	0.957	0.696	0.363	0.810	0.783
DoS	0.288	0.041	0.892	0.288	0.435	0.325	0.880	0.857
Worms	0.250	0.027	0.051	0.250	0.085	0.102	0.887	0.058
Weighted Avg.	0.592	0.324	0.730	0.592	0.552	0.341	0.848	0.819

Table 50: Confusion Matrix - Naïve Bayes

Reconnaissance	DoS	Worms	
2659	110	9	**Reconnaissance**
2186	946	157	**DoS**
23	4	9	**Worms**

The statistics of the Naïve Bayes classifier shows that it is not the best classifier for our task. The numbers of Table 49 demonstrate that the performance of this classifier is below the average and the fine-tuning and re-configuration it requires is substantial. The results of the confusion Matrix as follows (Table 50),

- The ideal Confusion Matrix should only have numbers in the yellow colour cells (Diagonal line).
- This classifier has labelled 2659 Reconnaissance attacks correctly and 119 incorrectly.
- It has spotted 946 DoS instances correctly. Regrettably, it has labelled 2186 DoS attacks as Reconnaissance and 157 DoS attacks as Worm attacks.
- Finally, this classifier manages to identify 9 Worm records correctly and it has missed 27 Worm records in the process of training.

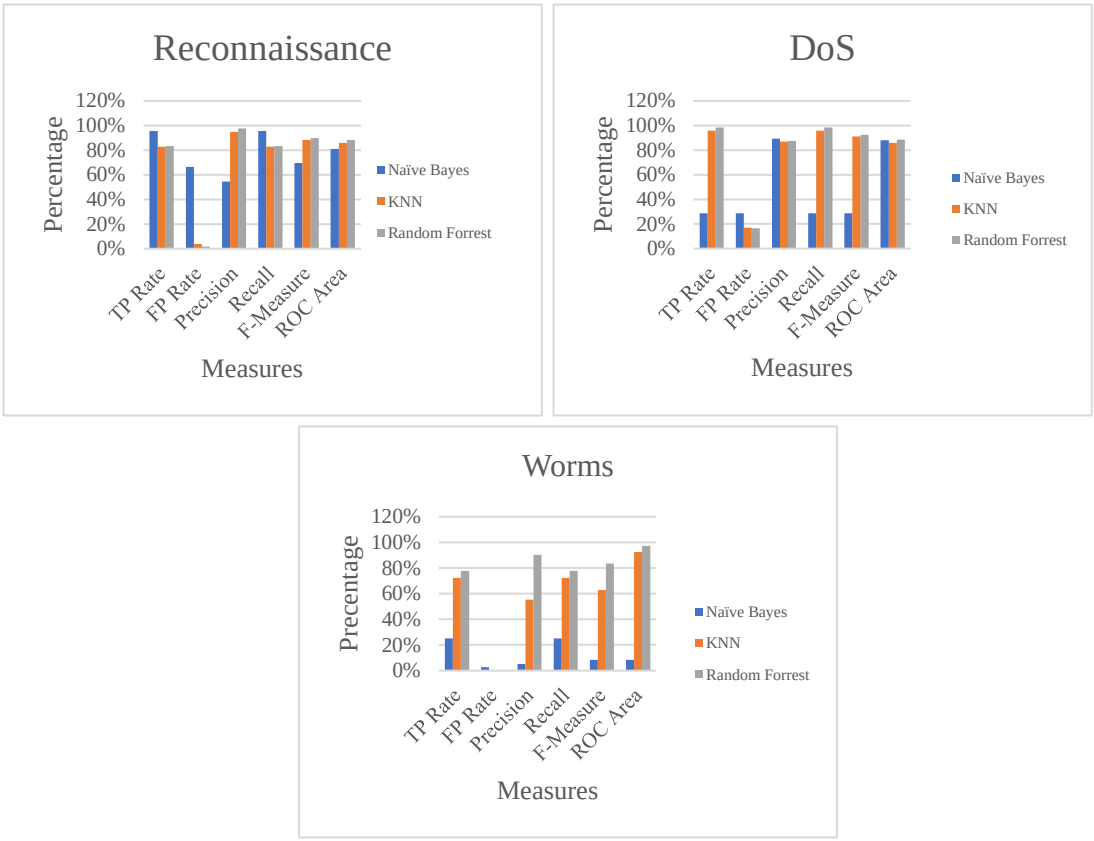


Figure 55: Classifier Performance Comparison

Figure 55 present a graphical view of each classifier's performances. Overall, the Random Forrest Classifier react favourably to our data set. This does not imply that we have a perfect model to detect Network Reconnaissance, DoS attacks or Worm Attacks. This model requires further adjustments to reach its true potential.

The above statistics highlight the importance of fine-tuning ML models to increase accuracy. Weka provides the facility to study the contribution of features towards the data set. The next section ranks 20 features based on the contribution to the data set. The complete Weka ranking output information is available in Appendix I: Weka – Ranking the Attributes (Features).

5.1.4 Discussion on Attribute Ranking

To rank the attributes using Weka; (in order to perform this procedure, the user has to load a data set to Weka): Open Weka Explorer → Click on Select Attributes → Choose Ranker from Search Method drop down → Click on start button.

Table 51: Ranked Attributes Using Weka

Features / Attribute	Score
smeansz	0.56742
sload	0.44216
proto	0.32187
dbytes	0.26841
dloss	0.23539
dload	0.20145
spkts	0.16972
ct_srv_dst	0.16889
ct_dst_sport_ltm	0.16257
ct_src_dport_ltm	0.14642
djit	0.14291
ct_dst_ltm	0.11531
ct_srv_src	0.10542
ct_src_ltm	0.09164
ct_state_ttl	0.08174
state	0.08157
dttl	0.07571
sttl	0.04983
service	0.00376
is_sm_ips_ports	0

As per Table 51, the Mean of the flow packet size transmitted by the source has the highest impact on the data set. Hence, the Weka has ranked smeanz feature on the top of the list. Features such as “Source bits per second” and the “Transaction protocol type” also influence the sample data set. Surprisingly, is_sm_ips_ports feature has no impact on the data set. The information in Table 51 is valuable for the ML model fine-tuning. As it quantifies the necessary values and presents them in ascending order. We will not discuss ML model fine-tuning process as it is out of this project’s scope.

5.2 Challenges and Limitations

In this section we will take you through the challenges and limitation we come across in our journey.

5.2.1 Installation and Configuration Process

We have included several applications to optimise the network anomaly detection in our NSM solution. It is important to state that all the applications in our framework are open-source. Also, these applications have a decent and blooming community working to improve the functionalities and usage. Freely available does not mean that it is easy to work with. Applications like Zeek, Elasticsearch, Filebeat, Kibana demands a good technical background and has a steep learning curve. These applications are not "plug and play" type application as it requires a bit of extra configuration and hours understand its functionalities.

Integrating Zeek with the EFK stack is not straight forward nor simple. It requires a lot of reading and testing. One could say that information should be available on the internet. It is rather complicated task, because the majority of the information sources are outdated and has a limited scope. In my case, I had to spend a lot of time figuring out issues in the Zeek and EFK integration process. I have documented the procedures step by step in the Appendices. This information may get change whenever a new version release with a different set of features. Installation, configuration, and deployment alone will do no good if users do not know how to work with the applications. I had to spend a considerable amount of time studying the functionalities prior doing the actual testing work.

5.2.2 EFK? Why Not ELK

I have replaced Logstash with Filebeat due to the challenges I have faced in the Zeek integration phase. Hence, the proposed stack is name as (E)lasticsearch, (F)ilebeat, (K)ibana and not (E)lasticsearch, (L)ogstach and (K)ibana. Both Filebeat and Logstash is used to dispatch log files from a file-based data source to a compatible output destination. Our purpose is not to develop an ELK stack but to improve the threat detection capabilities using the Zeek application. Hence, we have continued our work with the Filebeat application by ditching Logstash.

5.2.3 Storage Issues

An NSM system requires a large storage space to store the log files. This issue can be managed in a development environment or a non-production environment, but in a production environment, is it not ideal to have a small storage to store network logfiles. IT security specialist's advice to have a dedicated and secure space for storing log files in a production environment.

5.2.4 Integrating Machine Learning to Optimise the Detection Rate

As we have described in our previous chapters, our NSM solutions have two components. The first is to improve the Zeek capabilities to detect threats and the second is the apply machine learning technologies to further enhance the threat detection capabilities by identifying false-positives and false-negatives detection. This was a challenging task until I discover the Weka application. When learning and applying machine learning, there is so much to learn. For example:

- Understanding of the algorithms.
- Awareness on underlying mathematics required.
- I must learn and use specified tools and languages.

This requires time and resources, which I do not had when I start this project. Fortunately, the Weka application did not demand above requirements to perform the ML analysis

5.2.5 Insufficient Data to Train Machine Learning Model – Worm Attacks

Figure 49 clearly shows a drastic difference between the number of worm attacks and the number of DoS attacks and Reconnaissance activities. This is due to the data set limitations. The UNSW-NB15 only possess a limited number of Worm attacks information on the database. Hence, it is necessary to perform the test using a different data set with more information related to Worm arracks.

Next, I will list down the limitations of the proposed framework.

5.2.6 Log File Analysis

The Zeek produces a lot of log files, but we have only considered couples of them for this project. We can further enhance the threat detection capabilities by incorporating the rest of the log files generated by Zeek NSM.

5.2.7 Real-Time Threat Detection Using Machine Learning

As of now our system only perform passive Machine learning analysis. Moreover, our framework shows the way to analyse past log files. We are expecting to improve the framework to an active Machine learning analysis in our future work.

5.3 Conclusion

The focus of this project is to introduce a 3-tier approach, which is focused on improving the threat detection ability of a Network-based intrusion Detection System. We have conducted a comprehensive literature survey and study the existing network-based intrusion detection technologies and designed an efficient framework to improve the threat detection capabilities.

We have deployed a network security monitoring prototype using Zeek NSM, Elasticsearch, Filebeat and Kibana (EFK Stack), Weka application. The installation and configuration process is also clearly documented for readers to replicate our work if required. We have tested the proposed framework by testing the detection capabilities towards, DoS attacks, Reconnaissance attacks, Work attacks, NMAP scans. Finally, we have conducted a passive machine learning analysis using multiple machine learning algorithms.

Moreover, Decision Table, Decision Tree, Random Forrest, Naïve Bayes and K-Nearest Neighbors classifiers were used for the process of passive machine learning analysis. We have conceptually demonstrated and proved the capabilities of the proposed system and we outlined future research directions and the challenges we faced in the process of implementation in this dissertation work.

5.4 Future Work

Due to rapid growth of threats and challenges in Cybersecurity in it important to learn and do more experiments to enhance the network security defence capabilities. We also believe that open-source platforms such as Linux related applications should be more user friendly and should be used more often by non-technical personnel for their daily use. Having said that developers should also develop the applications accordingly. This is just the beginning of my journey in Network Threat Detection. Using the foundations laid in this project I would like to present potential future enhancement ideas.

- Introducing Zeek + EFK Framework for domestic use
    - The components we used in our framework are lightweight and free. I believe that this solution can be implemented in microcomputers such as Arduino/Raspberry pi and used to protect the home network from Cyber adversaries.
    - Users can plug the microcontroller into their home network and view the network behaviour using the web.
- Optimise the real-time threat detection using the Machine Learning technologies.
- Use complex machine learning algorithms for the model optimisation work.
    - In this project we have used Naïve bayes, KNN and Random Forrest classifiers. These classifiers are not the best fit for anomaly detection. There are complex algorithms available to detect anomalies. We are expecting to use such algorithms in our futures studies.
- Use different data set to train and test the framework.

This concludes our effort to develop a NSM system to detect threats. The following sections present information on the reference materials ,which we used for our project and the Appendices section. Appendices sections provide comprehensive instructions to build and configure the proposed environments.

REFERENCE LIST

[1] V. Chandola, A. Banerjee and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR),* vol. 41, no. 3, pp. 1-58, 2009.

[2] A. Lakhina, M. Crovella and C. Diot, "Diagnosing network-wide traffic anomalies," *ACM SIGCOMM computer communication review,* vol. 34, no. 4, pp. 219-230, 2004.

[3] N. Hoque, M. H. Bhuyan, R. C. Baishya, D. K. Bhattacharyya and J. K. Kalita, "Network attacks: Taxonomy, tools and systems," *Journal of Network and Computer Applications,* vol. 40, pp. 307-324, 2014.

[4] G. Fernandes, J. J. Rodrigues, L. F. Carvalho, J. F. Al-Muhtadi and M. L. Proena, "A comprehensive survey on network anomaly detection," *Telecommunication Systems,* vol. 70, no. 3, pp. 447-489, 2019.

[5] M. M. Ogonji, G. Okeyo and J. M. Wafula, "A survey on privacy and security of Internet of Things," *Computer Science Review,* vol. 38, 2020.

[6] S. Eggers, "A novel approach for analyzing the nuclear supply chain cyber-attack surface," *Nuclear Engineering and Technology,* 2020.

[7] M. Nieles, K. Dempsey and V. Pillitteri, "An introduction to information security," National Institute of Standards and Technology, 2017.

[8] M. Z. Gunduz and R. Das, "Cyber-security on smart grid: Threats and potential solutions," *Computer Networks,* vol. 169, 2020.

[9] F. Mouton, M. M. Malan and H. S. Venter, "Social engineering from a normative ethics perspective," in *2013 Information Security for South Africa*, 2013.

[10] R. A. Maxion and T. N. Townsend, "Masquerade detection using truncated command lines," in *Proceedings international conference on dependable systems and networks*, 2002.

[11] J. M. Vidal, A. L. S. Orozco and L. J. G. Villalba, "Online masquerade detection resistant to mimicry," *Expert Systems with Applications,* pp. 162-180, 2016.

[12] M. Feng and R. Gupta, "Detecting virus mutations via dynamic matching," in *2009 IEEE International Conference on Software Maintenance*, 2009.

[13] P. C. Van Oorschot, "Intrusion Detection and Network-Based Attacks," in *Computer Security and the Internet: Tools and Jewels*, Springer Nature, 2020, pp. 310-334.

[14] D. E. Denning, "An intrusion-detection model," *IEEE Transactions on software engineering,* no. 2, pp. 222-232, 1987.

[15] W. T. Yue and M. Cakanyildirim, "Intrusion prevention in information systems: Reactive and proactive responses," *Journal of Management Information Systems,* vol. 24, no. 1, pp. 329-353, 2007.

[16] A. Khraisat, I. Gondal, P. Vamplew and J. Kamruzzaman, "Survey of intrusion detection systems: techniques, datasets and challenges," *Cybersecurity,* vol. 2, no. 1, p. 20, 2019.

[17] M. M. Hassan, "Current studies on intrusion detection system, genetic algorithm and fuzzy logic," *International Journal of Distributed and Parallel Systems (IJDPS),* vol. 4, no. 2, 2013.

[18] H. Bostani and M. Sheikhan, "Hybrid of anomaly-based and specification-based IDS for Internet of Things using unsupervised OPF based on MapReduce approach," *Computer Communications,* pp. 52-71, 2017.

[19] C.-I. Fan, H.-W. Hsiao, C.-H. Chou and Y.-F. Tseng, "Malware detection systems based on API log data mining," in *2015 IEEE 39th annual computer software and applications conference*, 2015.

[20] P. Wang and Y.-S. Wang, "Malware behavioural detection and vaccine development by using a support vector model classifier," *Journal of Computer and System Sciences,* vol. 81, no. 6, pp. 1012-1026, 2015.

[21] J. B. Fraley and M. Figueroa, "Polymorphic malware detection using topological feature extraction with data mining," in *SoutheastCon 2016*, 2016.

[22] S. Gamage and J. Samarabandu, "Deep learning methods in network intrusion detection: A survey and an objective comparison," in *Journal of Network and Computer Applications*, 2020.

[23] K. Scarfone and P. Mell, "Guide to intrusion detection and prevention systems (idps)," National Institute of Standards and Technology, 2012.

[24] R. Sekar, A. Gupta, J. Frullo, T. Shanbhag, A. Tiwari, H. Yang and S. Zhou, "Specification-based anomaly detection: a new approach for detecting network intrusions," in *Proceedings of the 9th ACM conference on Computer and communications security,* 2002.

[25] M. V. de Assis, J. J. Rodrigues and M. L. Proenca Jr, "A seven-dimensional flow analysis to help autonomous network management," *Information Sciences,* vol. 278, pp. 900-913, 2014.

[26] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Macia-Fernandez and E. Vazquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges," *computers & security,* vol. 28, no. 1-2, pp. 18-28, 2009.

[27] S. Axelsson, “The base-rate fallacy and its implications for the difficulty of intrusion detection,” in *Proceedings of the 6th ACM Conference on Computer and Communications Security*, 1999.

[28] C. Kruegel, D. Mutz, W. Robertson and F. Valeur, “Bayesian event classification for intrusion detection,” in *19th Annual Computer Security Applications Conference, 2003. Proceedings*, 2003.

[29] S. Solani and N. K. Jadav, “A Novel Approach to Reduce False-Negative Alarm Rate in Network-Based Intrusion Detection System Using Linear Discriminant Analysis,” in *Inventive Communication and Computational Technologies*, 2020.

[30] C. Kruegel, F. Valeur, G. Vigna and R. Kemmerer, “Stateful intrusion detection for high-speed network's,” in *Proceedings 2002 IEEE Symposium on Security and Privacy*, 2002.

[31] J. E. Gaffney and J. W. Ulvila, “Evaluation of intrusion detectors: A decision theory approach,” in *Proceedings 2001 IEEE Symposium on Security and Privacy. S&P 2001*, 2000.

[32] S. J. Stolfo, W. Fan, W. Lee, A. Prodromidis and P. K. Chan, “Cost-based modeling for fraud and intrusion detection: Results from the JAM project,” in *Proceedings DARPA Information Survivability Conference and Exposition. DISCEX'00*, 2000.

[33] O. Bouziani, H. Benaboud, A. S. Chamkar and S. Lazaar, “A Comparative study of Open Source IDSs according to their Ability to Detect Attacks,” in *Proceedings of the 2nd International Conference on Networking, Information Systems & Security*, 2019.

[34] F. Siemons, “Open Source IDS: Snort or Suricata? [Updated 2021],” Infosec Institute, 27 January 2021. [Online]. Available: https://resources.infosecinstitute.com/topic/open-source-ids-snort-suricata/. [Accessed 21 May 2021].

[35] M. Schrotter, T. Scheffler and B. Schnor, “Evaluation of Intrusion Detection Systems in IPv6 Networks,” in *Proceedings of the 16th International Joint Conference on e-Business and Telecommunications (ICETE 2019)*, 2019.

[36] V. Paxson, “Bro: a system for detecting network intruders in real-time,” *Computer networks*, vol. 31, no. 23-24, pp. 2435-2463, 1999.

[37] S. Talukder, “Tools and techniques for malware detection and analysis,” 2020.

[38] M. Amrollahi, S. Hadayeghparast, H. Karimipour, F. Derakhshan and G. Srivastava, “Enhancing network security via machine learning: opportunities and challenges,” in *Handbook of Big Data Privacy*, Springer, 2020, pp. 165-189.

[39] R. J. Boyle and R. R. Panko, Corporate computer security, Pearson Upper Saddle River, 2013.

[40] . M. H. Bhuyan, D. K. Bhattacharyya and J. K. Kalita, “Networks and Network Traffic Anomalies,” in *Network traffic anomaly detection and prevention: concepts, techniques, and tools*, Springer, 2017, pp. 27-32.

[41] A. S. Ashoor and S. Gore, “Difference between intrusion detection system (IDS) and intrusion prevention system (IPS),” in *International Conference on Network Security and Applications*, 2011.

[42] R. Boutaba, M. A. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar, F. Estrada-Solano and O. M. Caicedo, “A comprehensive survey on machine learning for networking: evolution, applications and research opportunities,” *Journal of Internet Services and Applications*, vol. 9, no. 1, p. 16, 2018.

[43] S. Iqbal, M. L. M. Kiah, B. Dhaghighi, M. Hussain, S. Khan, M. K. Khan and K.-K. R. Choo, “On cloud security attacks: A taxonomy and intrusion detection and prevention as a service,” *Journal of Network and Computer Applications*, vol. 74, pp. 98-120, 2016.

[44] W. T. Yue and M. Cakanyildirim, “Intrusion prevention in information systems: Reactive and proactive responses,” *Journal of Management Information Systems*, vol. 24, no. 1, pp. 329-353, 2007.

[45] J. W. d. G. Stênico and L. L. Ling, “Network Traffic Monitoring,” in *The State of the Art in Intrusion Prevention and Detection*, Auerbach Publications, 2014, pp. 23-46.

[46] S. Bhatt, P. K. Manadhata and L. Zomlot, “The operational role of security information and event management systems,” *IEEE security & Privacy*, vol. 12, no. 5, pp. 35-41, 2014.

[47] O. Akinrolabu, I. Agrafiotis and A. Erola, “The challenge of detecting sophisticated attacks: Insights from SOC Analysts,” in *Proceedings of the 13th International Conference on Availability, Reliability and Security*, 2018.

[48] A. Milenkoski, M. Vieira, S. Kounev, A. Avritzer and B. D. Payne, “Evaluating computer intrusion detection systems: A survey of common practices,” *ACM Computing Surveys (CSUR)*, vol. 48, no. 1, pp. 1-41, 2015.

[49] Z. Jadidi, V. Muthukkumarasamy and E. Sithirasenan, “Artificial Intelligence-Based Intrusion Detection Techniques,” in *The State of the Art in Intrusion Prevention and Detection*, CRC Press, 2014, p. 285.

[50] C. Sekhar and K. V. Rao, “A Study: Machine Learning and Deep Learning Approaches for Intrusion Detection System,” in *International Conference on Computer Networks and Inventive Communication Technologies*, 2019.

[51] M. S. Husain, “Nature Inspired Approach for Intrusion Detection Systems,” *Design and Analysis of Security Protocol for Communication*, pp. 171-182, 2020.

[52] H. Jin, G. Xiang, F. Zhao, D. Zou, M. Li and L. Shi, “VMFence: A customized intrusion prevention system in distributed virtual computing environment,” in *3rd*

International Conference on Ubiquitous Information Management and Communication (ICUIMC'09), 2009.

[53] C. Modi and D. Patel, "A novel hybrid-network intrusion detection system (H-NIDS) in cloud computing," in *IEEE Symposium on Computational Intelligence in Cyber Security*, 2013.

[54] N. A. Azeez, T. M. Bada, S. Misra, A. Adewumi, C. V. d. Vyver and R. Ahuja, "Intrusion Detection and Prevention Systems: An Updated Review," in *Data Management, Analytics and Innovation*, Springer, 2020, pp. 685-696.

[55] CyberEdge Group, "2020 Cyberthreat Defence Report," CyberEdge Group, 2020.

[56] C. M. &. S. -. U. Department for Digital, "Cyber Security Breaches Survey 2020," *Computer Fraud & Security,* vol. 2020, no. 1361-3723, p. 4, 2020.

[57] C. M. &. S. Department for Digital, "Official Statistics Cyber Security Breaches Survey 2020," Department for Digital, Culture, Media & Sport, 26 March 2020. [Online]. Available: https://www.gov.uk/government/statistics/cyber-security-breaches-survey-2020/cyber-security-breaches-survey-2020#fn:1. [Accessed 10 February 2021].

[58] Sri Lanka Computer Emergency Readiness Team, "Sri Lanka Computer Emergency Readiness Team Annual Activity Report 2019," Sri Lanka Computer Emergency Readiness Team, 2019. [Online]. Available: https://www.cert.gov.lk/Downloads/General/Sri_Lanka_CERT_Annual_Activity_Report_2019.pdf. [Accessed 10 February 2021].

[59] A. H. Sung and S. Mukkamala, "Identifying important features for intrusion detection using support vector machines and neural networks," in *2003 Symposium on Applications and the Internet*, 2003.

[60] W. Lee and S. J. Stolfo, "A framework for constructing features and models for intrusion detection systems," *ACM transactions on Information and system security (TiSSEC),* vol. 3, no. 4, pp. 227-261, 2000.

[61] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *2010 IEEE symposium on security and privacy*, 2010.

[62] Z. Ahmad, A. S. Khan, C. Wai Shaiang, J. Abdullah and F. Ahmad, "Network intrusion detection system: A systematic study of machine learning and deep learning approaches," *Transactions on Emerging Telecommunications Technologies,* vol. 32, no. 1, p. 4150, 2020.

[63] Y. Meng, Y. Xiang and L.-F. Kwok, "Applications of Machine Learning in Intrusion Detection," in *The state of the art in intrusion prevention and detection*, CRC Press, 2014, pp. 311-332.

[64] H. Liu and B. Lang, “Machine learning and deep learning methods for intrusion detection systems: A survey,” *applied sciences*, vol. 9, no. 20, 2019.

[65] The Zeek Project, “Zeek Documentation current (v3.2.3),” The Zeek Project, 2019. [Online]. Available: https://docs.zeek.org/en/current/. [Accessed 11 February 2021].

[66] S. M. Sohi, J.-P. Seifert and F. Ganji, “RNNIDS: Enhancing network intrusion detection systems through deep learning,” *Computers & Security*, vol. 102, 2021.

[67] J. Grashöfer, C. Titze and H. Hartenstein, “Attacks on Dynamic Protocol Detection of Open Source Network Security Monitoring Tools,” in *2020 IEEE Conference on Communications and Network Security (CNS)*, 2020.

[68] C. V. Martinez, M. Sollfrank and B. Vogel-Heuser, “A Multi-Agent Approach for Hybrid Intrusion Detection in Industrial Networks: Design and Implementation,” in *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, 2019.

[69] L. Diederichsen, K.-K. R. Choo and N.-A. Le-Khac, “A graph database-based approach to analyze network log files,” in *International Conference on Network and System Security*, 2019.

[70] N. Moustafa and J. Slay, “UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set),” in *2015 military communications and information systems conference (MilCIS)*, 2015.

[71] C.-T. Yang, W.-J. Jiang, E. Kristiani, Y.-W. Chan and J.-C. Liu, “The Implementation of a Network Log System Using RNN on Cyberattack Detection with Data Visualization,” in *International Conference on Frontier Computing*, 2019.

[72] C.-K. Tsung, C.-T. Yang and S.-W. Yang, “Visualizing potential transportation demand from ETC log analysis using ELK stack,” *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6623-6633, 2020.

[73] E. B.V, “Filebeat overview,” Elasticsearch B.V, [Online]. Available: https://www.elastic.co/guide/en/beats/filebeat/current/filebeat-overview.html. [Accessed 18 03 2021].

[74] P. Sharma, K. Chaudhary, M. Wagner and M. G. M. Khan, “A comparative analysis of Malware anomaly detection,” in *Advances in Computer, Communication and Computational Sciences*, Springer, 2021, pp. 35-44.

[75] P. Gogoi, M. H. Bhuyan, D. Bhattacharyya and J. K. Kalita, “Packet and flow based network intrusion dataset,” in *International Conference on Contemporary Computing*, 2012.

[76] M. Ring, S. Wunderlich, D. Scheuring, D. Landes and A. Hotho, “A survey of network-based intrusion detection data sets,” *Computers & Security*, vol. 86, pp. 147-167, 2019.

[77] N. Moustafa and J. Slay, “The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set,” *Information Security Journal: A Global Perspective,* vol. 25, pp. 18-31, 2016.

[78] N. Moustafa and J. Slay, “The significant features of the UNSW-NB15 and the KDD99 data sets for network intrusion detection systems,” in *2015 4th international workshop on building analysis datasets and gathering experience returns for security (BADGERS)*, 2015.

[79] Verizon Communications Inc., “Verizon: 2019 Data Breach Investigations Report,” *Computer Fraud & Security,* 2019.

[80] J. Fruhlinger, “Top cybersecurity facts, figures and statistics,” CSO, 9 March 2020. [Online]. Available: https://www.csoonline.com/article/3153707/top-cybersecurity-facts-figures-and-statistics.html. [Accessed 22 May 2021].

[81] Cisco, “Cisco Annual Internet Report (2018–2023) White Paper,” Cisco, 9 March 2020. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html. [Accessed 22 May 2021].

[82] J. R. Vacca, “Chapter 6 - Intrusion Detection in Contemporary Environments,” in *Computer and Information Security Handbook (Third Edition)},,* Morgan Kaufmann, 2017, pp. 109-130.

[83] R. J. Bates, “Chapter 1 - Introduction,” in *Securing VoIP*, Syngress, 2015, pp. 1-34.

[84] M. Hao, “Analysis of the 2020 H1 Malware Trend,” NSFOCUS Technologies Group Co Ltd, 25 September 2020. [Online]. Available: https://nsfocusglobal.com/analysis-of-the-2020-h1-malware-trend/. [Accessed 22 May 2021].

[85] V. Benson and J. Mcalaney, “Chapter 4 - The social and psychological impact of cyberattacks,” in *Emerging Cyber Threats and Cognitive Vulnerabilities*, Academic Press, 2020, pp. 73-92.

[86] R. Patgiri, H. Katari, R. Kumar and D. Sharma, “Empirical study on malicious URL detection using machine learning,” in *International Conference on Distributed Computing and Internet Technology*, 2019.

APPENDICES

For this section we assume the readers have a rudimentary understanding of Linux commands & network configuration.

Appendix A: Zeek Installation & Configuration

In this section, we will take you through the installation and configuration of Zeek NSM step by step. In our scenario, we will be using the latest Ubuntu distribution (Ubuntu 20.04.2.0 desktop) for the installation. Ubuntu is a Debian based open-source Linux distribution. Hence, the following commands may not be similar on RPM/RedHat-based Linux, FreeBSD or macOS operating systems. Code segments that are highlighted in red denotes that the user should change the values according to his/her system settings.

a) Configure the Sniffing Interface.

Load Ubuntu .iso to a virtual box instance and install the image. Create a VirtualBox Host-Only Network Adapter. We are using **VirtualBox Host-Only Ethernet Adapter #3**. File → Host Network Manager → Create. The window of the Host Network Manager is depicted in Figure 56.

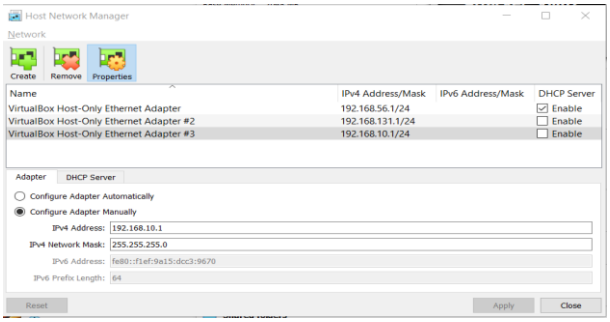


Figure 56: VirtualBox Host-Only Ethernet Adapter Setup

Next, setup the sniffing interface for **VirtualBox Host-Only Ethernet Adapter #3**. Settings → Network → Adapter 2 → Enable Network Adapter → Name: VirtualBox Host-Only Ethernet Adapter #3 → Advanced → Promiscuous Mode: Allow VMs → OK. Refer Figure 57 for the VM network setup.

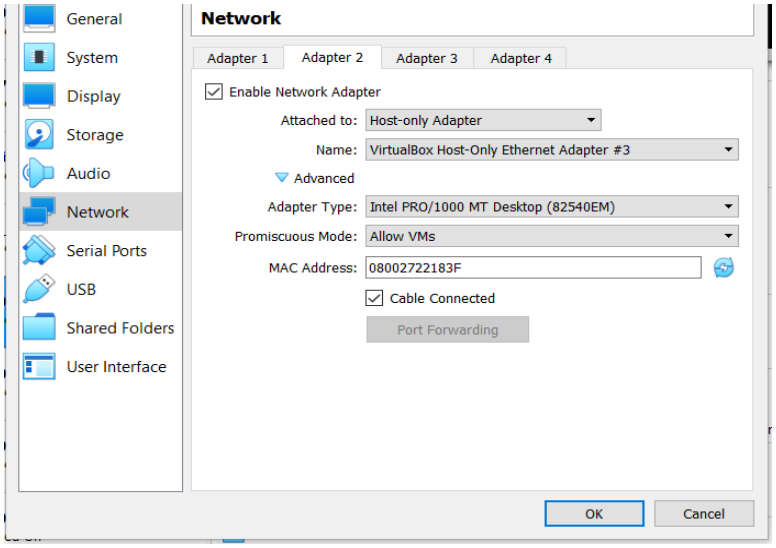


Figure 57: Setup the Sniffing Interface

Start the OS, identify the interface name. Open the terminal using Ctrl+Alt+T. Our interface is: **enp0s8.**

ifconfig

b) Install Dependencies Required for Zeek.

sudo su
apt-get update
apt-get upgrade
apt-get dist-upgrade
reboot
sudo apt-get install git
apt-get install curl
sudo apt install openjdk-8-jdk

sudo apt-get install cmake make gcc g++ flex bison libpcap-dev libssl-dev python3 python3-dev swig zlib1g-dev
sudo apt-get install python3-git python3-semantic-version
reboot

c) Installing Optional Dependencies

sudo apt search libmaxminddb
sudo apt-get install libmaxminddb-dev
apt-get tcpreplay
ip link list
ifconfig **enp0s8** mtu 16000
ip link list
/etc/init.d/networking restart
reboot
ip link list
reboot

d) Install Zeek NSM

- Make a directory for Zeek setup
  mkdir --parents **/home/nsm_vm_21/Downloads/zeek**
  cd **/home/nsm_vm_21/Downloads/zeek**

- Download zeek repository
  git clone --recursive https://github.com/zeek/zeek

- Configure Zeek
  cd zeek
  sudo mkdir --parents /opt/zeek
  ./configure --prefix=/opt/zeek/
  Make
  make install

- Set Zeek Path
  cp -p /etc/environment /etc/environment_backupcopy
  vi /etc/environment
  Add → :/opt/zeek/bin to the end
  Verify the modifications → echo $PATH
  reboot

e) Zeek Configuration

- Go to the installation directory.

  cd /opt/zeek/etc

- Edit the node.cfg file and enter the corresponding IP space.

  sudo vi node.cfg

  #[zeek]
  #type=standalone
  #host=securitynik
  #interface=enp0s25

  ## Below is an example clustered configuration. If you use this,
  ## remove the [zeek] node above.

  #[logger-1]
  #type=logger
  #host=localhost
  #
  [securitynik-zeek-manager]
  type=manager
  host=**192.168.10.7**
  #
  [securitynik-zeek-proxy]
  type=proxy
  host=**192.168.10.7**
  #
  [securitynik-zeek-worker-enp0s25]
  type=worker
  host=**192.168.10.7**
  interface=**enp0s8**
  #
  [securitynik-zeek-worker-lo]
  type=worker
  host=localhost
  interface=lo

  Save the file
  Shift+ZZ

- Edit the networks.cfg file and Enter the Corresponding IP Space.

  cd /opt/zeek/etc

  vi zeekctl.cfg
  change the Mail Option → MailTo = <**your email address**>

  Save the file
  Shift+ZZ
- Run Zeek

  sudo su
  cd /opt/zeek/bin
  sudo ./zeekctl
  install
  deploy
  Verify Zeek running by typing status

- Add Zeek Service to Systemd

  vi /etc/systemd/system/zeek.service

  Description=Zeek Network Security Monitoring

  Wants=network.target
  After=syslog.target network-online.target

  [Service]
  Type=simple
  ExecStart=zeekctl deploy
  Restart=on-failure
  RestartSec=10
  KillMode=process

  [Install]
  WantedBy=multi-user.target

  Save to file → Shift+ZZ

  cat /etc/systemd/system/zeek.service

- Next Reload System

  sudo systemctl daemon-reload

- Enable the Zeek Service
  sudo systemctl enable --now zeek.service

- Start the Service
  sudo systemctl start zeek.service

Appendix B: Elasticsearch Installation & Configuration

In this section, we will take you through the installation and configuration of Elasticsearch step by step. In our scenario, we will be using the latest Ubuntu distribution (Ubuntu 20.04.2.0 desktop) for the installation. Ubuntu is a Debian based open-source Linux distribution. Hence, the following commands may not be similar on RPM/RedHat-based Linux, FreeBSD or macOS operating systems. Code segments that are highlighted in red denotes that the user should change the values according to his/her system settings.

a) Download and Install the Public Signing Key.

```
sudo su
wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo apt-key add –
```

b) Install APT Repository

```
sudo apt-get install apt-transport-https
```

c) Download and Update the Repository

.

```
echo "deb https://artifacts.elastic.co/packages/7.x/apt stable main" | sudo tee /etc/apt/sources.list.d/elastic-7.x.list
```

d) Install Elasticsearch

```
sudo apt-get update && sudo apt-get install elasticsearch
```

e) Enable Journalctl Logging

```
cd /lib/systemd/system/
vi elasticsearch.service
```

cd /lib/systemd/system
vi elasticsearch.service

- Remove –quite option
- Shift+ZZ

sudo /bin/systemctl daemon-reload
sudo /bin/systemctl enable elasticsearch.service
sudo journalctl --unit elasticsearch

f) Configure Elasticsearch to Start Automatically.

sudo /bin/systemctl daemon-reload

g) Enable the Elasticsearch.Service

sudo /bin/systemctl enable elasticsearch.service

h) Start the Elasticsearch service

sudo systemctl start elasticsearch.service

i) Verify Elastic Programme Running in the System.

systemctl status elasticsearch

Appendix C: Filebeat Installation & Configuration

In this section, we will take you through the installation and configuration of Filebeat step by step. In our scenario, we will be using the latest Ubuntu distribution (Ubuntu 20.04.2.0 desktop) for the installation. Ubuntu is a Debian based open-source Linux distribution. Hence, the following commands may not be similar on RPM/RedHat-based Linux, FreeBSD or macOS operating systems. Code segments that are highlighted in red denotes that the user should change the values according to his/her system settings.

a) Install Filebeat

```
sudo apt-get update && sudo apt-get install filebeat
```

b) Configure Filebeat to Run on Startup

```
sudo update-rc.d filebeat defaults 95 10
reboot
```

c) Start Filebeat Service

```
sudo systemctl start filebeat
sudo systemctl status filebeat
```

Appendix D: Kibana Installation & Configuration

In this section, we will take you through the installation and configuration of Kibana step by step. In our scenario, we will be using the latest Ubuntu distribution (Ubuntu 20.04.2.0 desktop) for the installation. Ubuntu is a Debian based open-source Linux distribution. Hence, the following commands may not be similar on RPM/RedHat-based Linux, FreeBSD or macOS operating systems. Code segments that are highlighted in red denotes that the user should change the values according to his/her system settings.

a) Install Kibana

```
sudo apt-get update && sudo apt-get install kibana
```

b) Enable Kibana Service

```
sudo /bin/systemctl daemon-reload
sudo /bin/systemctl enable kibana.service
```

c) Start Kibana Service

```
sudo systemctl start kibana.service
sudo systemctl status kibana.service
```

Appendix E: Zeek-EFL Integration

a) Kibana Configuration

Subsequent changes are depicted in Figure 58.

sudo cd /etc/kibana
sudo vi kibana.yml

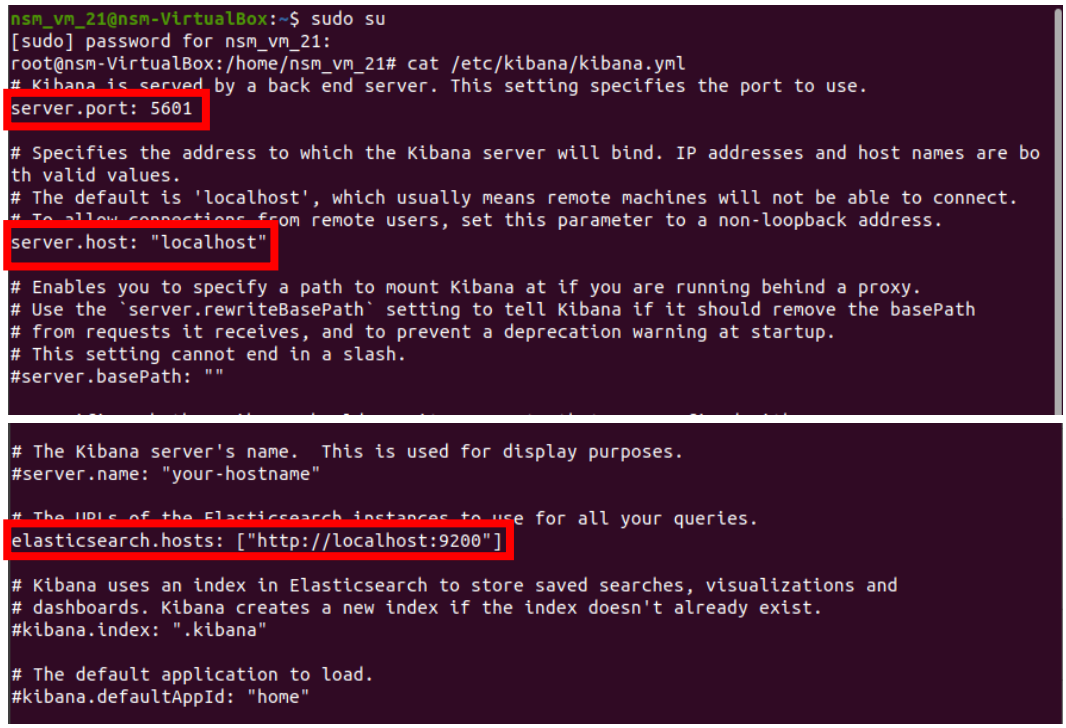


Figure 58: Kibana Configuration - Kibana.yml

b) Elasticsearch Configuration

Corresponding changes are depicted in Figure 59

cd /etc/elasticsearch
sudo vi elasticsearch.yml

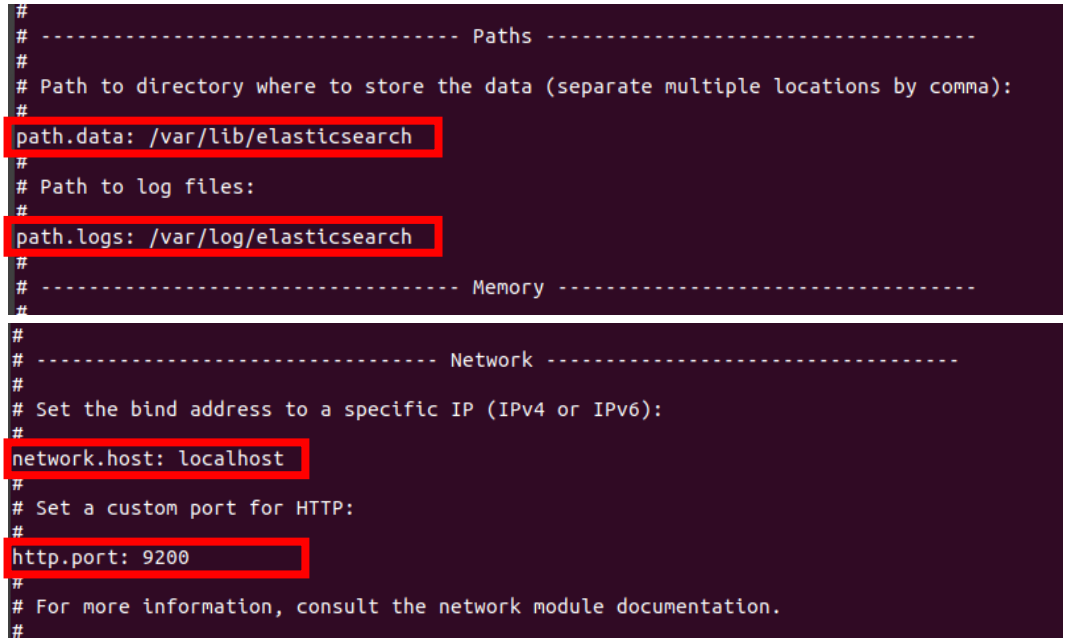


Figure 59: Elasticsearch Configuration - Elasticsearch.yml

c) Filebeat Configuration

Subsequent changes are portrayed in Figure 61 to Figure 62.

cd /etc/filebeat/

sudo vi filebeat.yml

```
# ============================== Filebeat modules ==============================

filebeat.config.modules:
  # Glob pattern for configuration loading
  path: ${path.config}/modules.d/*.yml

  # Set to true to enable config reloading
  reload.enabled: false

  # Period on which files under path should be checked for changes
  #reload.period: 10s

# ======================= Elasticsearch template setting =======================

setup.template.settings:
  index.number_of_shards: 1
  #index.codec: best_compression
  #_source.enabled: false


# ================================== General ===================================

# The name of the shipper that publishes the network data. It can be used to group
# all the transactions sent by a single shipper in the web interface.
name: zeek-beat

# The tags of the shipper are included in their own field with each
# transaction published.
tags: ["zeek"]

# Optional fields that you can specify to add additional information to the
# output.
#fields:
#  env: staging
```

```
# ================================== Dashboards ================================
# These settings control loading the sample dashboards to the Kibana index. Loading
# the dashboards is disabled by default and can be enabled either by setting the
# options here or by using the `setup` command.
setup.dashboards.enabled: true
setup.dashboards.directory: ${path.home}/kibana
setup.dashboards.beat: filebeat

# The URL from where to download the dashboards archive. By default this URL
# has a value which is computed based on the Beat name and version. For released
# versions, this URL points to the dashboard archive on the artifacts.elastic.co
# website.
#setup.dashboards.url:

# =================================== Kibana ===================================

# Starting with Beats version 6.0.0, the dashboards are loaded via the Kibana API.
# This requires a Kibana endpoint configuration.
setup.kibana:

  # Kibana Host
  # Scheme and port can be left out and will be set to the default (http and 5601)
  # In case you specify and additional path, the scheme is required: http://localhost:5601/path
  # IPv6 addresses should always be defined as: https://[2001:db8::1]:5601
  host: "localhost:5601"

  # Kibana Space ID
  # ID of the Kibana Space into which the dashboards should be loaded. By default,
  # the Default Space will be used.
  #space.id:
```

Figure 60:Filebeat Configuration ii - Filebeat.yml

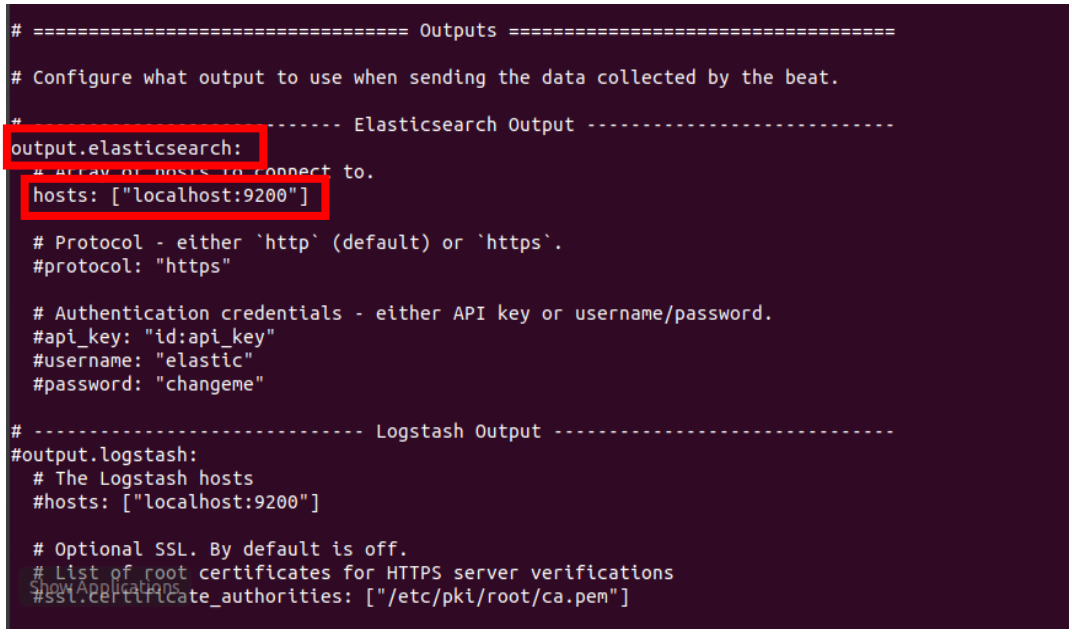


Figure 62:Filebeat Configuration iii - Filebeat.yml

cd modules.d
filebeat modules enable zeek

Enter sudo vi zeek.yml to edit the log files as shown in Figure 63.

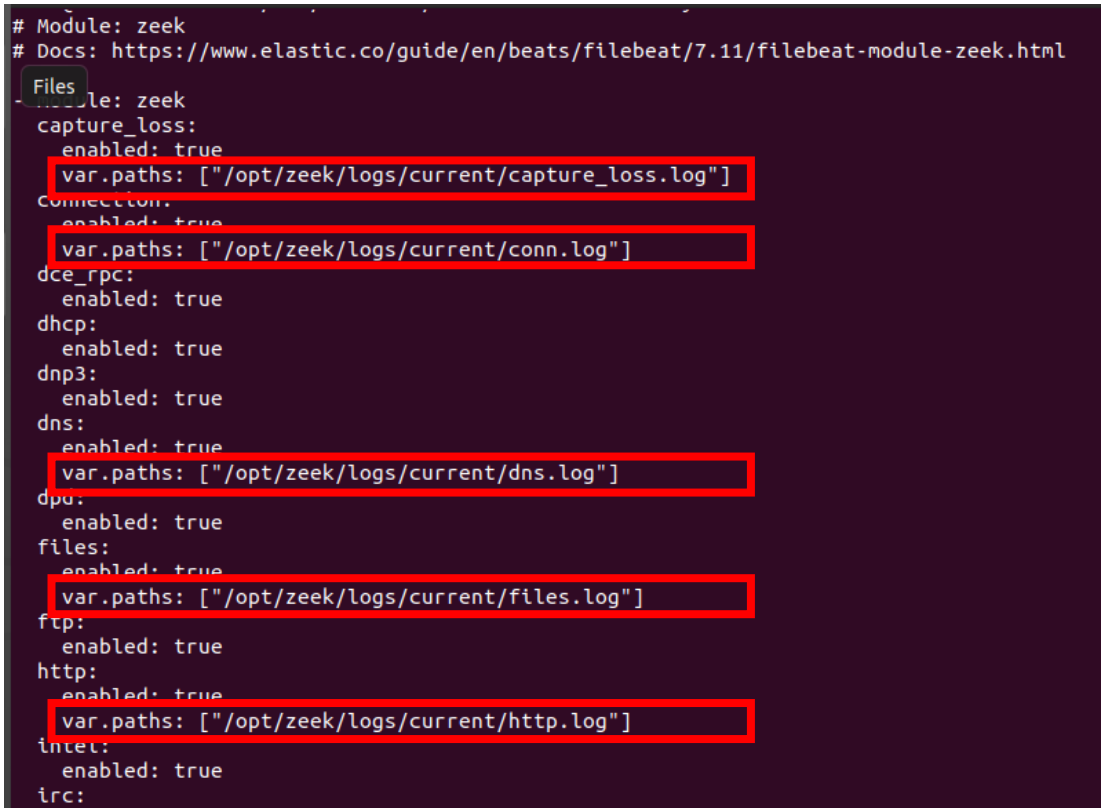


Figure 63: zeek.yml File Configuration

d) Restart All Services.

System status information are illustrated in Figure 64.

systemctl restart zeek
systemctl restart elasticsearch
systemctl restart kibana
systemctl restart filebeat

systemctl status zeek
systemctl status elasticsearch
systemctl status kibana
systemctl status filebeat

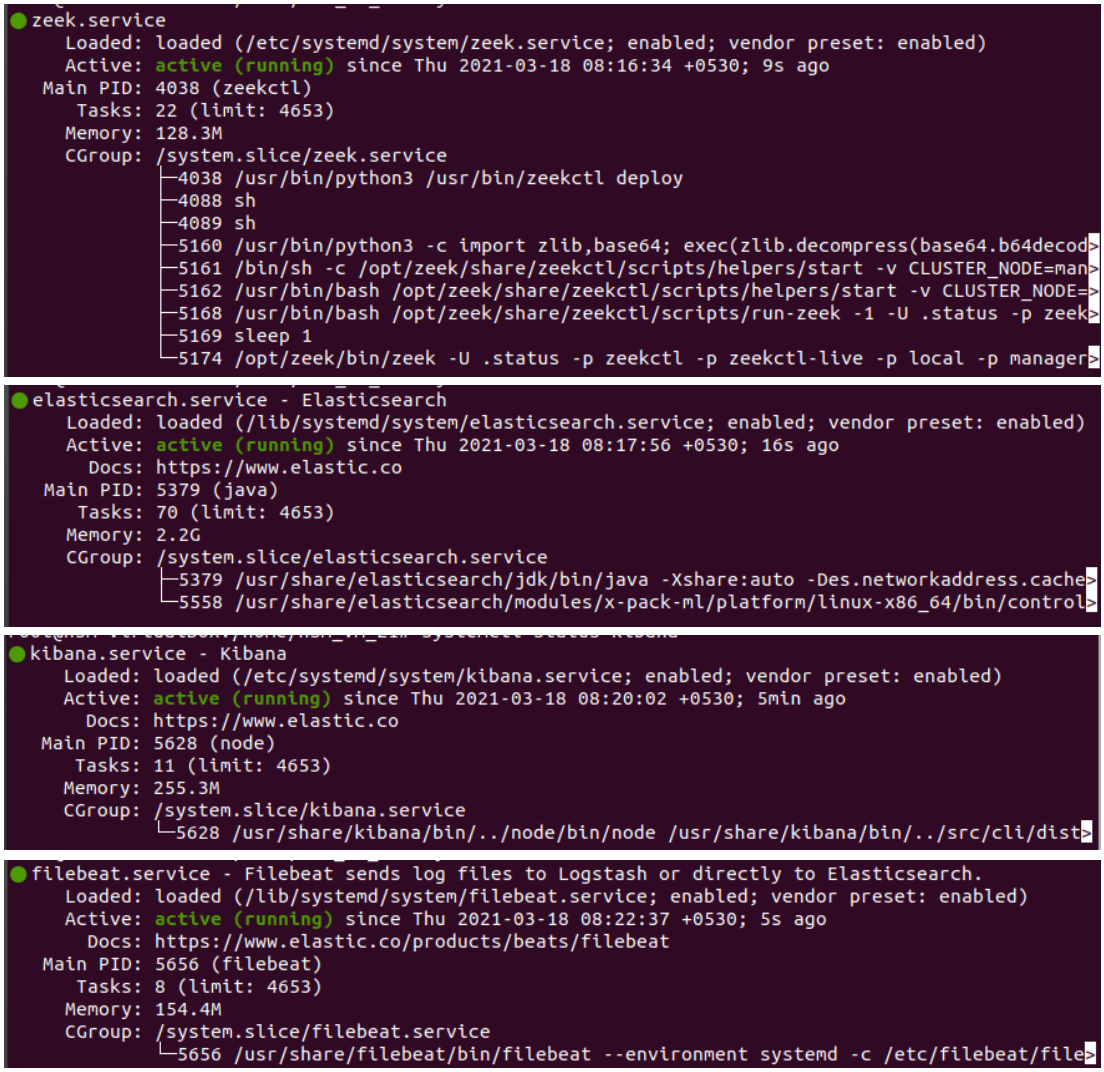


Figure 64: Checking the application status

e) Explore Kibana Dashboard

Type 127.0.0.1:5601 in your web browser (Figure 65)

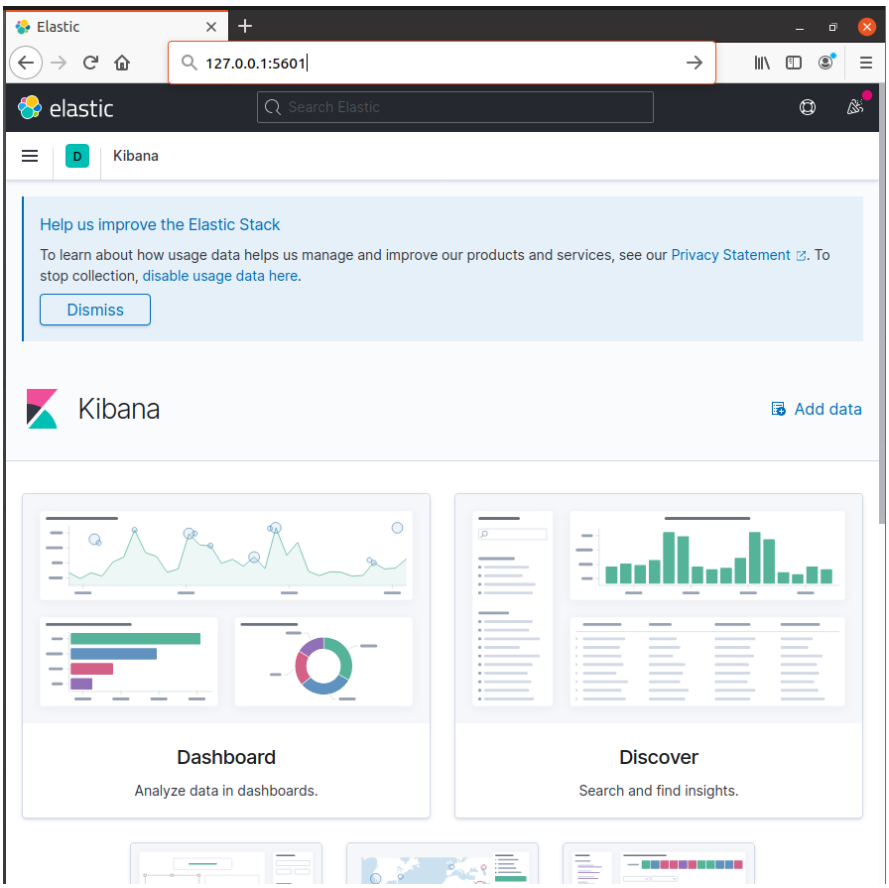


Figure 65: Accessing Kibana

f) Setup filebeat service in Kibanna

Discover → Manage spaces → Index patters →Search for filebear-* → Create index pattern → Select @timestamp from the drop down → Create index patterns.

Appendix F: Weka Installation

a) Download Weka application

Go to the download page https://sourceforge.net/projects/weka/files/weka-3-8/3.8.5/weka-3-8-5-azul-zulu-linux.zip/download?use_mirror=liquidtelecom and download weka to your NSM system.
sudo su
Go to the download directory.
Extract the file. Right click weka-3-8-5-azul-zulu-linux.zip → Extract here/Extract to.
cd weka-3-8-5-azul-zulu-linu
cd weka-3-8-5

b) Run Weka application

java -jar weka.jar

Above command invoke Weka GUI(Figure 66).

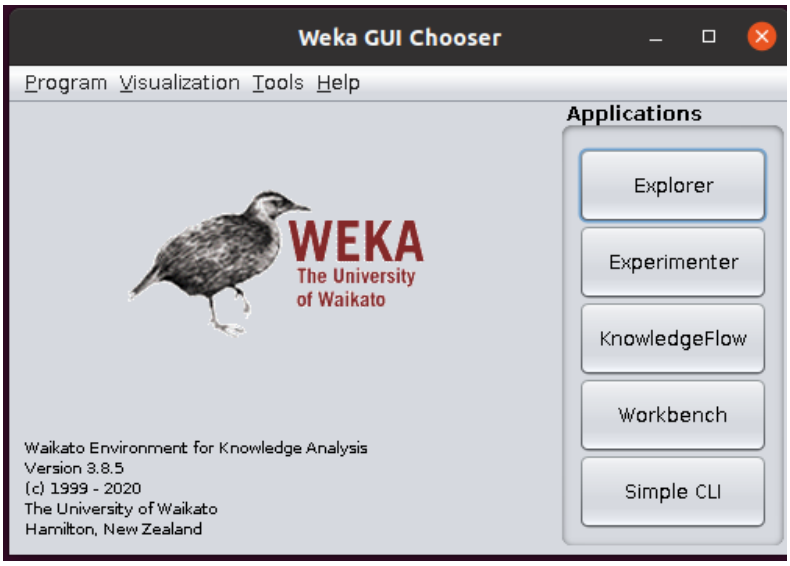


Figure 66: Weka GUI

Appendix G: Data Pre-processing Process

In this section we take you through the Linux command we used for the file format conversion. This process used in our data pre-processing procedure. The main objective of the data pre-processing procedure is to convert the data set into Weka readable format.

a) Convert .tsv file to .csv file format and using Linux utilities.

cat **/home/nsm_vm_21/Desktop/Scripts/conn.log** | zeek-cut **ts id.orig_h id.resp_h id.orig_p id.resp_p proto** > **temp_a_1.csv**
cat **temp_a_1.csv** | tr '[\t]' '[,]' > **temp_a_2.csv**
sed 's/\.//g' **temp_a_2.csv** > **temp_a_3**.csv
sed 's/-/?/g' **temp_a_3.csv** > **temp_a_4.csv**

cat **/home/nsm_vm_21/Desktop/Scripts/conn.log** | zeek-cut **ts id.orig_h id.resp_h id.orig_p id.resp_p proto** > **temp_b_1**.csv
cat **temp_b_1.csv** | tr '[\t]' '[,]' > **temp_b_2.csv**
sed 's/\.//g' **temp_b_2.csv** > **temp_b_3.csv**
sed 's/-/?/g' **temp_b_3.csv** > **temp_b_4.csv**

b) Label the .csv file

awk '{print $0 ",1"}' **temp_a_4.csv** > **Malicious.csv**
awk '{print $0 ",0"}' **temp_b_4.csv** > **Normal.csv**

c) Develop trainset.arff and testset.arff

cat **Malicious.csv Normal.csv** > **Netflow_temp.csv**
shuf **Netflow_temp.csv** > **Netflow.csv**
head -n **400 Netflow.csv** > **Testset.csv**
sed -e '1,300d' **Netflow.csv** > **trainset.arff**
sed 's/.$/?/' **Testset.csv** > **testset.arff** ##remove labels

Appendix H: Weka – Classifier Outputs

This section presents Weka output information of the classifiers, which were used for ML training

a) Results of the Model Training - Random Forrest Classifier

```
=== Run information ===

Scheme:       weka.classifiers.trees.RandomForest -P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S
1
Relation:     UNSW-NB15-Processed
Instances:    30514
Attributes:   21
              ï»¿proto
              state
              dbytes
              sttl
              dttl
              dloss
              service
              sload
              dload
              spkts
              smeansz
              djit
              is_sm_ips_ports
              ct_state_ttl
              ct_srv_src
              ct_srv_dst
              ct_dst_ltm
              ct_src_ltm
              ct_src_dport_ltm
              ct_dst_sport_ltm
              Class
Test mode:    split 80.0% train, remainder test

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

RandomForest
Bagging with 100 iterations and base learner

weka.classifiers.trees.RandomTree -K 0 -M 1.0 -V 0.001 -S 1 -do-not-check-capabilities

Time taken to build model: 9.02 seconds
=== Evaluation on test split ===
Time taken to test model on test split: 0.46 seconds
```

=== Summary ===

Correctly Classified Instances	5577	91.3813 %
Incorrectly Classified Instances	526	8.6187 %
Kappa statistic	0.8263	
Mean absolute error	0.1055	
Root mean squared error	0.2429	
Relative absolute error	31.5028 %	
Root relative squared error	59.3596 %	
Total Number of Instances	6103	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0.833	0.017	0.976	0.833	0.899	0.834	0.883	0.922	Reconnaissance
	0.983	0.166	0.874	0.983	0.925	0.834	0.884	0.808	DoS
	0.778	0.000	0.903	0.778	0.836	0.837	0.972	0.886	Worms
Weighted Avg.	0.914	0.097	0.921	0.914	0.913	0.834	0.884	0.860	

=== Confusion Matrix ===

a	b	c		<-- classified as
2315	462	1	|	a = Reconnaissance
53	3234	2	|	b = DoS
3	5	28	|	c = Worms

b) Results of the Model Training - KNN Classifier

=== Run information ===

Scheme: weka.classifiers.lazy.IBk -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""
Relation: UNSW-NB15-Processed
Instances: 30514
Attributes: 21
ï»¿proto
state
dbytes
sttl
dttl
dloss
service
sload
dload
spkts
smeansz
djit
is_sm_ips_ports
ct_state_ttl
ct_srv_src
ct_srv_dst
ct_dst_ltm
ct_src_ltm

```
        ct_src_dport_ltm
        ct_dst_sport_ltm
        Class
Test mode:    split 80.0% train, remainder test

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

IB1 instance-based classifier
using 1 nearest neighbour(s) for classification


Time taken to build model: 0 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 7.76 seconds

=== Summary ===

Correctly Classified Instances        5477               89.7427 %
Incorrectly Classified Instances      626                10.2573 %
Kappa statistic                       0.7942
Mean absolute error                   0.1074
Root mean squared error                  0.2629
Relative absolute error               32.0486 %
Root relative squared error           64.2532 %
Total Number of Instances             6103

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure  MCC      ROC Area  PRC Area  Class
               0.828     0.038     0.947      0.828     0.884      0.803    0.859     0.890     Reconnaissance
               0.958     0.170     0.869      0.958     0.911      0.801    0.859     0.780     DoS
               0.722     0.003     0.553      0.722     0.627      0.630    0.924     0.403     Worms
Weighted Avg.  0.897     0.109     0.903      0.897     0.897      0.801    0.860     0.828

=== Confusion Matrix ===

   a     b     c      <-- classified as
 2300   473     5   |  a = Reconnaissance
  122  3151    16   |  b = DoS
    6     4    26   |  c = Worms
```

c) Results of the Model Training - Naïve Bayes Classifier

```
=== Run information ===

Scheme:       weka.classifiers.bayes.NaiveBayes
Relation:     UNSW-NB15-Processed
Instances:    30514
Attributes:   21
        ï»¿proto
        state
```

dbytes
sttl
dttl
dloss
service
sload
dload
spkts
smeansz
djit
is_sm_ips_ports
ct_state_ttl
ct_srv_src
ct_srv_dst
ct_dst_ltm
ct_src_ltm
ct_src_dport_ltm
ct_dst_sport_ltm
Class

Test mode: split 80.0% train, remainder test

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

Naive Bayes Classifier

Attribute	Class Reconnaissance (0.46)	DoS (0.54)	Worms (0.01)
ï»¿proto			
tcp	6966.0	3337.0	154.0
udp	4891.0	528.0	22.0
ospf	230.0	945.0	1.0
sctp	114.0	427.0	1.0
sep	18.0	84.0	1.0
swipe	16.0	81.0	1.0
sun-nd	16.0	81.0	1.0
mobile	16.0	81.0	1.0
pim	16.0	81.0	1.0
ip	8.0	45.0	1.0
ipnip	8.0	45.0	1.0
ggp	8.0	45.0	1.0
st2	8.0	45.0	1.0
egp	8.0	45.0	1.0
cbt	8.0	45.0	1.0
emcon	8.0	45.0	1.0
igp	8.0	45.0	1.0
xnet	8.0	45.0	1.0
nvp	8.0	45.0	1.0
argus	8.0	45.0	1.0
bbn-rcc	8.0	45.0	1.0
chaos	8.0	45.0	1.0
pup	8.0	45.0	1.0

hmp	8.0		45.0	1.0
mux	8.0		45.0	1.0
dcn	8.0		45.0	1.0
prm	8.0		45.0	1.0
trunk-1	8.0		45.0	1.0
xns-idp	8.0		45.0	1.0
trunk-2	8.0		45.0	1.0
leaf-1	8.0		45.0	1.0
leaf-2	8.0		45.0	1.0
irtp	8.0		45.0	1.0
rdp	8.0		45.0	1.0
iso-tp4	8.0		45.0	1.0
netblt	8.0		45.0	1.0
mfe-nsp	8.0		45.0	1.0
merit-inp	8.0		45.0	1.0
3pc	8.0		45.0	1.0
idpr	8.0		45.0	1.0
xtp	8.0		.0	1.0
tp++	8.0		45.0	1.0
ddp	8.0		45.0	1.0
idpr-cmtp	8.0		45.0	1.0
ipv6	15.0		85.0	1.0
il	8.0		45.0	1.0
idrp	8.0		45.0	1.0
ipv6-frag	8.0		45.0	1.0
sdrp		8.0	45.0	1.0
ipv6-route		8.0	45.0	1.0
gre		20.0	108.0	1.0
rsvp		15.0	89.0	1.0
mhrp		8.0	45.0	1.0
bna		8.0	45.0	1.0
i-nlsp		8.0	45.0	1.0
narp		8.0	45.0	1.0
ipv6-no		8.0	45.0	1.0
tlsp		8.0	45.0	1.0
skip		8.0	45.0	1.0
ipv6-opts		8.0	45.0	1.0
any		22.0	133.0	1.0
cftp		8.0	45.0	1.0
sat-expak		8.0	45.0	1.0
kryptolan		8.0	45.0	1.0
rvd		8.0	45.0	1.0
ippc		8.0	45.0	1.0
sat-mon		8.0	45.0	1.0
ipcv		8.0	45.0	1.0
visa		8.0	45.0	1.0
cpnx		8.0	45.0	1.0
cphb		8.0	45.0	1.0
wsn		8.0	45.0	1.0
pvp		8.0	45.0	1.0
br-sat-mon		8.0	45.0	1.0
wb-mon		8.0	45.0	1.0
wb-expak		8.0	45.0	1.0

iso-ip	8.0	45.0	1.0
secure-vmtp	8.0	45.0	1.0
vmtp	8.0	45.0	1.0
vines	8.0	45.0	1.0
ttp	8.0	45.0	1.0
nsfnet-igp	8.0	45.0	1.0
dgp	8.0	45.0	1.0
tcf	8.0	45.0	1.0
eigrp	8.0	45.0	1.0
sprite-rpc	8.0	45.0	1.0
larp	8.0	45.0	1.0
mtp	8.0	45.0	1.0
ax.25	8.0	45.0	1.0
ipip	8.0	45.0	1.0
micp	8.0	45.0	1.0
aes-sp3-d	8.0	45.0	1.0
encap	8.0	45.0	1.0
etherip	8.0	45.0	1.0
pri-enc	8.0	45.0	1.0
gmtp	8.0	45.0	1.0
pnni	8.0	45.0	1.0
ifmp	8.0	45.0	1.0
aris	8.0	45.0	1.0
qnx	8.0	45.0	1.0
a/n	8.0	45.0	1.0
scps	8.0	45.0	1.0
compaq-peer	8.0	45.0	1.0
ipcomp	8.0	45.0	1.0
snp	8.0	45.0	1.0
ipx-n-ip	8.0	45.0	1.0
vrrp	8.0	45.0	1.0
zero	8.0	45.0	1.0
pgm	8.0	45.0	1.0
iatp	8.0	45.0	1.0
ddx	8.0	45.0	1.0
l2tp	8.0	45.0	1.0
srp	8.0	45.0	1.0
stp	8.0	45.0	1.0
smp	8.0	45.0	1.0
uti	8.0	45.0	1.0
sm	8.0	45.0	1.0
ptp	8.0	45.0	1.0
crtp	8.0	45.0	1.0
isis	8.0	45.0	1.0
fire	8.0	45.0	1.0
crudp	8.0	45.0	1.0
sccopmce	8.0	45.0	1.0
pipe	8.0	45.0	1.0
sps	8.0	45.0	1.0
iplt	8.0	45.0	1.0
unas	841.0	5247.0	1.0
fc	8.0	45.0	1.0
ib	8.0	45.0	1.0

[total]	14116.0	16482.0	303.0

state

FIN	6957.0	3238.0	152.0
INT	6904.0	12440.0	22.0
CON	59.0	408.0	3.0
REQ	68.0	269.0	1.0
CLO	2.0	1.0	1.0
RST	3.0	3.0	1.0
[total]	13993.0	16359.0	180.0

dbytes

mean	1626.4821	21536.7112	76554.5562
std. dev.	95770.0673	413977.9667	200747.6277
weight sum	13987	16353	174
precision	9977.8972	9977.8972	9977.8972

sttl

mean	254.0763	234.3127	251.092
std. dev.	14.1667	56.8509	25.4773
weight sum	13987	16353	174
precision	85	85	85

dttl

mean	126.2739	53.3858	222.4655
std. dev.	126.2826	101.9888	82.4189
weight sum	13987	16353	174
precision	84.3333	84.3333	84.3333

dloss

mean	0.6031	7.9628	30.0782
std. dev.	35.3417	154.4133	78.0464
weight sum	13987	16353	174
precision	39.0567	39.0567	39.0567

service

http	2074.0	1710.0	149.0
snmp	81.0	20.0	1.0
pop3	1.0	88.0	1.0
ftp-data	1.0	15.0	1.0
ftp	1.0	74.0	1.0
smtp	8.0	403.0	1.0
ssl	1.0	9.0	1.0
irc	1.0	8.0	1.0
dns	48.0	148.0	1.0
dhcp	1.0	43.0	1.0
[total]	2217.0	2518.0	158.0

sload

mean	70452936.9112	123335795.9398	72677428.4277
std. dev.	122525276.3116	187239907.8647	278775249.86
weight sum	13987	16353	174
precision	477130.7179	477130.7179	477130.7179

dload			
mean	2426.0625	17008.1225	132754.3814
std. dev.	24584.0373	120486.9763	301770.0603
weight sum	13987	16353	174
precision	207.2277	207.2277	207.2277
spkts			
mean	1.1256	21.508	10.6445
std. dev.	15.0084	244.0848	26.5865
weight sum	13987	16353	174
precision	37.0427	37.0427	37.0427
smeansz			
mean	78.5719	139.1523	183.4845
std. dev.	38.755	176.05	223.2945
weight sum	13987	16353	174
precision	2.1293	2.1293	2.1293
djit			
mean	107.3576	118.6089	397.4429
std. dev.	209.7589	867.6906	623.4995
weight sum	13987	16353	174
precision	4.4605	4.4605	4.4605
is_sm_ips_ports			
mean	0	0	0
std. dev.	0.0017	0.0017	0.0017
weight sum	13987	16353	174
precision	0.01	0.01	0.01
ct_state_ttl			
mean	1.5156	1.8318	1.1092
std. dev.	0.5965	0.7225	0.3468
weight sum	13987	16353	174
precision	1	1	1
ct_srv_src			
mean	2.6589	4.5078	1.6613
std. dev.	2.5099	3.3261	0.9645
weight sum	13987	16353	174
precision	1.1471	1.1471	1.1471
ct_srv_dst			
mean	2.1873	4.4609	1.5726
std. dev.	2.1022	3.2841	1.204
weight sum	13987	16353	174
precision	1.375	1.375	1.375
ct_dst_ltm			
mean	1.8584	2.6511	1.6207
std. dev.	2.4004	2.0858	2.0487
weight sum	13987	16353	174

precision	1.175	1.175	1.175
ct_src_ltm			
mean	2.705	4.2999	2.6149
std. dev.	4.5113	5.1071	4.5871
weight sum	13987	16353	174
precision	1	1	1
ct_src_dport_ltm			
mean	0.386	1.6943	0.1839
std. dev.	1.103	1.9926	0.6217
weight sum	13987	16353	174
precision	2.2857	2.2857	2.2857
ct_dst_sport_ltm			
mean	1.5004	2.339	1.3448
std. dev.	0.7279	1.4767	0.2372
weight sum	13987	16353	174
precision	1.3	1.3	1.3

Time taken to build model: 0.11 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 0.1 seconds

=== Summary ===

Correctly Classified Instances	3614	59.2168 %
Incorrectly Classified Instances	2489	40.7832 %
Kappa statistic	0.2491	
Mean absolute error	0.2717	
Root mean squared error	0.5126	
Relative absolute error		81.0873 %
Root relative squared error		125.2745 %
Total Number of Instances	6103	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0.957	0.664	0.546	0.957	0.696	0.363	0.810	0.783	Reconnaissance
	0.288	0.041	0.892	0.288	0.435	0.325	0.880	0.857	DoS
	0.250	0.027	0.051	0.250	0.085	0.102	0.887	0.058	Worms
Weighted Avg.	0.592	0.324	0.730	0.592	0.552	0.341	0.848	0.819	

=== Confusion Matrix ===

a	b	c		<-- classified as
2659	110	9	|	a = Reconnaissance
2186	946	157	|	b = DoS
23	4	9	|	c = Worms

Appendix I: Weka – Ranking the Attributes (Features)

=== Run information ===

Evaluator: weka.attributeSelection.InfoGainAttributeEval
Search: weka.attributeSelection.Ranker -T -1.7976931348623157E308 -N -1
Relation: UNSW-NB15-Processed
Instances: 30514
Attributes: 21
ï»¿proto
state
dbytes
sttl
dttl
dloss
service
sload
dload
spkts
smeansz
djit
is_sm_ips_ports
ct_state_ttl
ct_srv_src
ct_srv_dst
ct_dst_ltm
ct_src_ltm
ct_src_dport_ltm
ct_dst_sport_ltm
Class
Evaluation mode: evaluate on all training data

=== Attribute Selection on all input data ===

Search Method:
Attribute ranking.

Attribute Evaluator (supervised, Class (nominal): 21 Class):
Information Gain Ranking Filter

Ranked attributes:
0.56742 11 smeansz
0.44216 8 sload
0.32187 1 ï»¿proto
0.26841 3 dbytes
0.23539 6 dloss
0.20145 9 dload
0.16972 10 spkts
0.16889 16 ct_srv_dst
0.16257 20 ct_dst_sport_ltm
0.14642 19 ct_src_dport_ltm
0.14291 12 djit

0.11531 17 ct_dst_ltm
0.10542 15 ct_srv_src
0.09164 18 ct_src_ltm
0.08174 14 ct_state_ttl
0.08157 2 state
0.07571 5 dttl
0.04983 4 sttl
0.00376 7 service
0 13 is_sm_ips_ports

Selected attributes: 11,8,1,3,6,9,10,16,20,19,12,17,15,18,14,2,5,4,7,13 :

Appendix J: Weka – Decision Table Predictive Model Output Information

The testset.arff contained 300 records. Due to a formatting issue, we have removed a record from the input file. Hence, the following output contains 299 instances.

=== Re-evaluation on test set ===

User supplied test set
Relation: networktraffic
Instances: unknown (yet). Reading incrementally
Attributes: 8

=== Predictions on user test set ===

inst#	actual	predicted	error	prediction
1	1:?	1:1		0.999
2	1:?	2:0		0.998
3	1:?	1:1		0.999
4	1:?	2:0		0.998
5	1:?	1:1		0.999
6	1:?	1:1		0.999
7	1:?	1:1		0.999
8	1:?	2:0		0.998
9	1:?	2:0		0.998
10	1:?	1:1		0.999
11	1:?	2:0		0.998
12	1:?	2:0		0.998
13	1:?	2:0		0.998
14	1:?	1:1		0.999
15	1:?	2:0		0.998
16	1:?	1:1		0.999
17	1:?	1:1		0.999
18	1:?	1:1		0.999
19	1:?	1:1		0.999
20	1:?	2:0		0.998
21	1:?	2:0		0.998
22	1:?	1:1		0.999
23	1:?	2:0		0.998
24	1:?	2:0		0.998
25	1:?	1:1		0.999
26	1:?	2:0		0.998
27	1:?	1:1		0.999
28	1:?	1:1		0.999
29	1:?	2:0		0.998
30	1:?	1:1		0.999
31	1:?	1:1		0.999
32	1:?	1:1		0.999
33	1:?	2:0		0.998
34	1:?	1:1		0.999
35	1:?	2:0		0.998

36	1:?	1:1	0.999
37	1:?	1:1	0.999
38	1:?	1:1	0.999
39	1:?	1:1	0.999
40	1:?	1:1	0.999
41	1:?	2:0	0.998
42	1:?	2:0	0.998
43	1:?	1:1	0.999
44	1:?	1:1	0.999
45	1:?	1:1	0.999
46	1:?	1:1	0.999
47	1:?	1:1	0.999
48	1:?	2:0	0.998
49	1:?	1:1	0.999
50	1:?	2:0	0.998
51	1:?	1:1	0.999
52	1:?	1:1	0.999
53	1:?	2:0	0.998
54	1:?	2:0	0.998
55	1:?	1:1	0.999
56	1:?	1:1	0.999
57	1:?	1:1	0.999
58	1:?	1:1	0.999
59	1:?	1:1	0.999
60	1:?	2:0	0.998
61	1:?	2:0	0.998
62	1:?	2:0	0.998
63	1:?	1:1	0.999
64	1:?	2:0	0.998
65	1:?	1:1	0.999
66	1:?	1:1	0.999
67	1:?	2:0	0.998
68	1:?	1:1	0.999
69	1:?	2:0	0.998
70	1:?	1:1	0.999
71	1:?	2:0	0.998
72	1:?	1:1	0.999
73	1:?	1:1	0.999
74	1:?	1:1	0.999
75	1:?	2:0	0.998
76	1:?	1:1	0.999
77	1:?	1:1	0.999
78	1:?	1:1	0.999
79	1:?	1:1	0.999
80	1:?	1:1	0.999
81	1:?	2:0	0.998
82	1:?	1:1	0.999
83	1:?	1:1	0.999
84	1:?	1:1	0.999
85	1:?	2:0	0.998
86	1:?	1:1	0.999
87	1:?	2:0	0.998
88	1:?	1:1	0.999

89	1:?	2:0	0.998
90	1:?	1:1	0.999
91	1:?	2:0	0.998
92	1:?	1:1	0.999
93	1:?	1:1	0.999
94	1:?	2:0	0.998
95	1:?	1:1	0.999
96	1:?	2:0	0.998
97	1:?	1:1	0.999
98	1:?	2:0	0.998
99	1:?	1:1	0.999
100	1:?	2:0	0.998
101	1:?	2:0	0.998
102	1:?	1:1	0.999
103	1:?	1:1	0.999
104	1:?	1:1	0.999
105	1:?	1:1	0.999
106	1:?	2:0	0.998
107	1:?	1:1	0.999
108	1:?	1:1	0.999
109	1:?	1:1	0.999
110	1:?	1:1	0.999
111	1:?	1:1	0.999
112	1:?	1:1	0.999
113	1:?	1:1	0.999
114	1:?	1:1	0.999
115	1:?	1:1	0.999
116	1:?	1:1	0.999
117	1:?	1:1	0.999
118	1:?	1:1	0.999
119	1:?	1:1	0.999
120	1:?	2:0	0.998
121	1:?	1:1	0.999
122	1:?	1:1	0.999
123	1:?	2:0	0.998
124	1:?	1:1	0.999
125	1:?	2:0	0.998
126	1:?	2:0	0.998
127	1:?	2:0	0.998
128	1:?	1:1	0.999
129	1:?	2:0	0.998
130	1:?	1:1	0.999
131	1:?	2:0	0.998
132	1:?	1:1	0.999
133	1:?	1:1	0.999
134	1:?	1:1	0.999
135	1:?	1:1	0.999
136	1:?	1:1	0.999
137	1:?	2:0	0.998
138	1:?	1:1	0.999
139	1:?	2:0	0.998
140	1:?	2:0	0.998
141	1:?	1:1	0.999

142	1:?	2:0	0.998
143	1:?	2:0	0.998
144	1:?	2:0	0.998
145	1:?	1:1	0.999
146	1:?	1:1	0.999
147	1:?	2:0	0.998
148	1:?	1:1	0.999
149	1:?	1:1	0.999
150	1:?	1:1	0.999
151	1:?	2:0	0.998
152	1:?	1:1	0.999
153	1:?	1:1	0.999
154	1:?	2:0	0.998
155	1:?	1:1	0.999
156	1:?	2:0	0.998
157	1:?	2:0	0.998
158	1:?	1:1	0.999
159	1:?	2:0	0.998
160	1:?	2:0	0.998
161	1:?	1:1	0.999
162	1:?	1:1	0.999
163	1:?	1:1	0.999
164	1:?	2:0	0.998
165	1:?	2:0	0.998
166	1:?	1:1	0.999
167	1:?	2:0	0.998
168	1:?	1:1	0.999
169	1:?	1:1	0.999
170	1:?	1:1	0.999
171	1:?	2:0	0.998
172	1:?	1:1	0.999
173	1:?	1:1	0.999
174	1:?	1:1	0.999
175	1:?	2:0	0.998
176	1:?	1:1	0.999
177	1:?	1:1	0.999
178	1:?	2:0	0.998
179	1:?	1:1	0.999
180	1:?	1:1	0.999
181	1:?	1:1	0.999
182	1:?	1:1	0.999
183	1:?	2:0	0.998
184	1:?	2:0	0.998
185	1:?	2:0	0.998
186	1:?	1:1	0.999
187	1:?	1:1	0.999
188	1:?	2:0	0.998
189	1:?	2:0	0.998
190	1:?	1:1	0.999
191	1:?	1:1	0.999
192	1:?	1:1	0.999
193	1:?	1:1	0.999
194	1:?	1:1	0.999

195	1:?	1:1	0.999
196	1:?	1:1	0.999
197	1:?	1:1	0.999
198	1:?	1:1	0.999
199	1:?	2:0	0.998
200	1:?	2:0	0.998
201	1:?	2:0	0.998
202	1:?	1:1	0.999
203	1:?	2:0	0.998
204	1:?	2:0	0.998
205	1:?	1:1	0.999
206	1:?	1:1	0.999
207	1:?	1:1	0.999
208	1:?	1:1	0.999
209	1:?	2:0	0.998
210	1:?	1:1	0.999
211	1:?	2:0	0.998
212	1:?	1:1	0.999
213	1:?	2:0	0.998
214	1:?	1:1	0.999
215	1:?	2:0	0.998
216	1:?	2:0	0.998
217	1:?	1:1	0.999
218	1:?	1:1	0.999
219	1:?	2:0	0.998
220	1:?	1:1	0.999
221	1:?	1:1	0.999
222	1:?	1:1	0.999
223	1:?	1:1	0.999
224	1:?	1:1	0.999
225	1:?	1:1	0.999
226	1:?	1:1	0.999
227	1:?	1:1	0.999
228	1:?	2:0	0.998
229	1:?	1:1	0.999
230	1:?	1:1	0.999
231	1:?	1:1	0.999
232	1:?	2:0	0.998
233	1:?	1:1	0.999
234	1:?	1:1	0.999
235	1:?	1:1	0.999
236	1:?	1:1	0.999
237	1:?	1:1	0.999
238	1:?	2:0	0.998
239	1:?	2:0	0.998
240	1:?	1:1	0.999
241	1:?	2:0	0.998
242	1:?	1:1	0.999
243	1:?	2:0	0.998
244	1:?	1:1	0.999
245	1:?	2:0	0.998
246	1:?	2:0	0.998
247	1:?	1:1	0.999

248	1:?	1:1	0.999
249	1:?	2:0	0.998
250	1:?	2:0	0.998
251	1:?	1:1	0.999
252	1:?	1:1	0.999
253	1:?	1:1	0.999
254	1:?	1:1	0.999
255	1:?	1:1	0.999
256	1:?	2:0	0.998
257	1:?	1:1	0.999
258	1:?	2:0	0.998
259	1:?	2:0	0.998
260	1:?	1:1	0.999
261	1:?	2:0	0.998
262	1:?	2:0	0.998
263	1:?	1:1	0.999
264	1:?	1:1	0.999
265	1:?	1:1	0.999
266	1:?	2:0	0.998
267	1:?	1:1	0.999
268	1:?	2:0	0.998
269	1:?	1:1	0.999
270	1:?	1:1	0.999
271	1:?	2:0	0.998
272	1:?	1:1	0.999
273	1:?	1:1	0.999
274	1:?	2:0	0.998
275	1:?	1:1	0.999
276	1:?	1:1	0.999
277	1:?	1:1	0.999
278	1:?	2:0	0.998
279	1:?	2:0	0.998
280	1:?	1:1	0.999
281	1:?	1:1	0.999
282	1:?	1:1	0.999
283	1:?	1:1	0.999
284	1:?	1:1	0.999
285	1:?	2:0	0.998
286	1:?	1:1	0.999
287	1:?	1:1	0.999
288	1:?	2:0	0.998
289	1:?	1:1	0.999
290	1:?	2:0	0.998
291	1:?	1:1	0.999
292	1:?	1:1	0.999
293	1:?	1:1	0.999
294	1:?	1:1	0.999
295	1:?	2:0	0.998
296	1:?	1:1	0.999
297	1:?	1:1	0.999
298	1:?	1:1	0.999
299	1:?	2:0	0.998

=== Summary ===

Total Number of Instances	0	
Ignored Class Unknown Instances		299