2022

A Prototype for predicting network intrusions using Bayesian based networks.

Kirui, Pamela Chemutai
School of Computing and Engineering Sciences
Strathmore University

A Prototype for Predicting Network Intrusions Using Bayesian Based Networks

By

Kirui, Pamela Chemutai

122849

A Dissertation Submitted to the School of Computing and Engineering Sciences in partial fulfilment of the requirements for the award of Masters of Science in Information Systems Security

Masters of Science in Information Systems Security

Strathmore University

December 2021

DECLARATION

I declare that this work has not been previously submitted and approved for the award of a degree by this or any other University. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made in the thesis itself.

Chemutai, Pamela Kirui

1st December, 2021

APPROVAL

The thesis of Chemutai Pamela Kirui was reviewed and approved by the following:

Dr. Humphrey Njogu,
Senior Lecturer, Faculty of Information Technology
Strathmore University

Dr. Joseph Orero
Dean, Faculty of Information Technology
Strathmore University

Prof. Ruth Kiraka,
Dean, School of Graduate Studies,
Strathmore University

ACKNOWLEDGEMENT

First, I thank God for good health and strength.

My sincere gratitude to my supervisor, Dr. Humphrey Njogu whose advice and constant guidance has shaped this dissertation. Many thanks to Dr Vincent Omwenga and Dr Joseph Sevilla for their support throughout the coursework.

I thank you all most sincerely.

DEDICATION

A special dedication to my late dad Mathews Kirui

My mum Mary Kirui

My dear brothers Abdalla and Patrick

And

My sisters Risperth and Patricia

ABSTRACT

As technology advances, networks get more sophisticated. As a result, the attack surface for hackers has continued to expand leading to rapid increase in insecurity; therefore, there is need for a line of defence that is reactive and predictive. Traditional protection techniques such as data encryption, user authentication and avoiding programming errors are in existence and act as the first line of defence for computer security however, these techniques are not sufficient to protect against malicious code and insider attacks. Attacks such as programming errors are unavoidable due to the complexity of the system and application software that is rapidly evolving and consequently leaves behind some weaknesses that could be exploited.

Research on Intrusion Detection Systems (IDS) has been considered a critical research area to bridge this gap .The challenge with network-based detection is the ability to scheme the behaviour of normal and abnormal traffic. This calls for a reliable system that can learn the structure of network data and differentiate between normal and abnormal. Since there are many applications using different internet protocols an IDS finds it difficult to detect all kinds of attacks efficiently. It suffers from the difficulty of building robust schemes that result in increasing false alarm rates caused by weak feature selection, inefficient classifier generation and data noise generated from imbalanced data. Due to this, predictive Machine language (ML) algorithms have been proposed since they are capable of solving such problems.

Various ML methods have previously been employed in areas of network intrusion detections however; Bayesian based Network has been considered a better approach due to its significant features. In this study, experiments were carried out using KDD99 data set. The first experiment was conducted using Weka, a Machine Learning tool and the second experiment was conducted using Python language. First, the data went through pre-processing where most relevant features were selected from the entire data set before classification and thereafter issues of data noise such as class imbalance were removed. Naive Bayes, a Bayesian based Network was used as a classifier to train and test the data in Weka. Secondly, Python language was used to train and test the classifier. In both experiments, training and testing ratios were 0.67 and 0.33 respectively. The algorithm obtained accuracy of 92% using Weka tool and of 90% using Python (JupyterLab)

Keywords: Network Intrusion Detection; Naïve Bayes.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

FTP- File Transfer Protocol

HTTP- Hypertext Transfer Protocol

POP-Post Office Protocol

U2R-User To Root

ARFF- Attribute-Relation File Format

AUC- Area Under The Curve

CSV- Comma-Separated Values

DoS- Denial Of Service

HIDS -Host-Based Intrusion Detection Systems

IDS- Intrusion Detection Systems

IPS- Intrusion Prevention Systems

NB - Naïve Bayes

NIDS- Network-Based Intrusion Detection Systems (Nids)

R2L- Remote To Local

ROC - Receiver Operating Characteristic

SMTP- Simple Mail Transfer Protocol

SNMP- Simple Network Management Protocol

SVM - Support Vector Machine

TCP- Transmission Control Protocol

Weka-Waikato Environment For Knowledge Analysis

CHAPTER 1: INTRODUCTION

1.1 Background of the study

There is tremendous growth of network-based services due to its advantages. Nowadays internet is extensively used, with its growth rising from one billion users in 2007 to over five billion internet users as of March 2021 ("Internet world stats - Usage and population statistics," n.d.) Due to this rapid growth of internet use, number of intrusions to computer systems through the network has immensely increased and the attacks continue to get more sophisticated(Allen, Christie & McHugh 2000).It has led to increased risks of attacks by intruders making the internet and the network unsafe for users. Network security is now of more importance than ever before and has become a serious issue for both private users and cooperate users.

Existing techniques used to detect network intrusions include anti-spam, anti-virus, firewalls, Intrusion Detection Systems and Intrusion Prevention Systems. Anti-spam helps in rejecting or blocking an email spam while Anti-virus identifies, blocks and eliminates malicious software (Fan, Yao & Zhao, 2006). A firewall is a defence, which acts as a filter for network traffic by using a set of rules. It prevents hosts from outside the internal network to connect to a secure end system. However, they suffer from the ability to keep track of the processes; a persistent attacker can easily bypass a firewall and gain entry into the network. Therefore there was a necessity for an Intrusion Detection System to detect possible malicious actions that bypass the firewalls and automatically learn the structure of network and as a result an IDS was considered as a second defence line to protect computer systems (Fan, Yao & Zhao, 2006)

Detection of intrusions can be from various available sources within the computer system or networks and can revolve around the host or around the network. Based on the source type, IDS can be categorized as Host-based Intrusion Detection Systems (HIDS) or Network-based Intrusion Detection Systems (NIDS).NIDS monitors traffic that travels through a network segment such as internet packets entering a network, if there is any anomaly it either raises an alarm, creates a log or respond based on how the systems is designed(Kumaravel,2016).HIDS monitors activities taking place on the system (Sarmin & Vasumathi, 2017).It monitors such activities done by host, operating systems logs, application logs then it alerts in case of suspicious system activities. Another category of IDS is based on scope by looking for instances of malicious behaviour or looking for specific signatures. These two strategies of IDS

are used to detect signs of network intrusions (Markou & Singh, 2003). They are classified as signature based and anomaly based IDSs. Signature based IDS compares the signature of an incoming connection to the signatures of previous attacks that are stored in a database (Singh, 2016). The problem with this approach is that if the database is not updated, the system will fail to identify new attack unless it is trained. Anomaly detection attempts to identify behaviour that deviates from the normal network. This type of detection is adaptive and can deal with new attack, however due to dynamic and sophisticated nature of attacks it is difficult to identify the specific type of attack anomaly.

Intrusion Detection Systems are mostly software based and monitor the network activity by detecting potential attacks and malicious activities in networks (Bace & Mell, 2001). IDS detects an intrusion and sends an alert to the system (Allah, Tabash & Tawfik, 2019).It is difficult to implement an IDS due to dynamic changes of intrusion which get more intelligent, hence a flexible IDS which is self adjustable and able to keep up with the changes is required. To address these challenges experienced by conventional IDSs, machine learning techniques have been proposed for intrusion detection in the recent past(Wang,2009).Machine learning techniques have been proven capable of detecting new attacks that were previously unknown or unseen(Silverstone,2003)

However, Machine Learning approaches can have shortcomings of increasing false alarm rates, which is attributed to weak feature selection, inefficient classifier generation, data noise and learning from imbalanced data (Napolitano, Khoshgoftaar & VanHulse, 2010).These classification methods require many training samples to build the classification scheme and accurate intrusion detection remains a challenge.

A Bayesian based network is preferred due to its ability to obtain coherent result from probabilistic information about a situation. This scheme encodes dependencies among all variables. It handles situations appropriately such as missing values. When classifying instances, the attribute with missing values is ignored and not included during probability calculation. This reduces the data noise resulting from imbalanced data and hence provides maximum classification accuracy, which is effective in situation prediction.

The focus of the study is to apply a probabilistic scheme for network intrusion detection using Bayesian Networks.

1.2 Problem Statement

In recent years, internet use has rapidly grown and currently 5.1 billion people in the world are using the internet with a proportional increase in networked machine, which has led to increased network intrusions. Intrusion activities are not only from external attackers but internal attackers too are gaining popularity abusing their privileges for their personal interest ("FBI: Internet crime report 2020," 2021). There are various solutions to curb intrusions and IDS is one of the techniques widely used in this field.

Intrusion detection is considered as the first line of defence in detecting an attack and most researchers have explored Bayesian network approach due its unique way of handling problems based on uncertainty. Bayesian networks' being a probabilistic technique handles a given consequence using a conditional probability formula. It predicts a certain type of attack given some observed system events, a feature that is suitable for finding anomaly in the network by the process of comparing patterns of data with known attack patterns to detect attacks with known behaviour. However, this causes an imbalance since non-rare events that are dominant, are used for training the classifier hence it overwhelms the rare events in the dataset, a challenge of Bayesian classifiers that is still being experienced by most researchers.

1.3 General Objective

The overall objective of this study is to develop a prototype for Network intrusion detection using Bayesian Networks.

1.4 Specific Objectives

1. To identify common network threats, vulnerabilities and attacks.
2. To review existing approaches, tools and techniques used in Network intrusion detection.
3. To design, develop and test a prototype based on Bayesian Networks for Network intrusion detection.
4. To validate the effectiveness of Bayesian Network-based Intrusion detection prototype.

1.5 Research Questions

1. What are the common types of network threats, vulnerabilities and attacks?

2. What are the existing tools, approaches and techniques used for network intrusion detection?
3. How can Bayesian Networks based prototype be designed, developed and tested for Network intrusion detection?
4. How can Bayesian network prototype be validated for effectiveness?

1.6 Justification

In today's society, most people can access a networked machine and this has led to increased network intrusions such as DDoS attack, Ransom ware attack and Botnet attacks. Many crucial and valuable assets are prone to Intrusion activities not only from external attackers but also internal attackers who are gaining popularity by abusing their privileges for personal interest("FBI: Internet crime report 2020," 2021).Hence there is need to protect these assets by monitoring the systems against attacks. Some anomalies have become so dynamic and complicated as hackers present intrusions in different ways, which may result to loss of valuable assets. The study will make it possible to discover new ideas that will consequently improve and build on previous research conducted especially on anomaly attacks in order to provide reliable automated systems that are able to continuously monitor network traffic for both normal attacks and anomalies .Therefore it is important to deploy reliable intelligent defense mechanisms that are intelligent to respond to intrusions in a quick and accurate manner.

A Bayesian based network is an ideal solution due to its potential to improve performance by reducing false alerts, increasing speed, computational time and accuracy

1.7 Scope and Limitation

This study focuses on detection of Network intrusions using Bayesian based networks.

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

The chapter reviews relevant research areas to deepen understanding of what Intrusion detection entails. The chapter provides an overview of computer networks as well as discusses various threats, vulnerabilities and attacks of computer networks. The chapter exhaustively discusses on Intrusion Detection Systems (IDS) how it works, how they are classified and then followed by an analysis of Machine learning approaches used to solve intrusion detection and finally a presentation of a conceptual framework.

2.2 Computer Networks

A computer network consists of two or more computers connected through wire or wireless medium that allows computers to transmit, exchange and share data and resources (Lowe, 2005).A computer network is built using hardware such as routers, switches, access points, and cables and software's such as operating systems or business applications (IBM, 2019)

Computer networks are linked by nodes like computers, routers and switches by use of cables, fibre optics, or wireless signals. These connections allow devices to communicate, share information and resources in a network.

2.2.1 Importance of computer networks

Computer networks are important in the growth of any business whether large or small; it has a great effect on improving the efficiency of the communication system and application requirements. Key task of computer network is to connect people, support applications and services and provide accessibility to resources that help the business to run smoothly. When all computers are networked it improves storage efficiency since there will be combined storage volume of all computers in the network. Due to its efficiency and flexibility, individuals, business applications and government institutions now use computer networks.

Businesses use computers to share resources. A company may have a computer for every worker and use them to style products, write brochures, and do the payroll. Initially, a number of these computers may have worked in isolation from the others, but at some

point, management may have decided to attach them to be ready to distribute information throughout the company (Tanenbaum & Wetherall, 2010).The goal is to form all programs, equipment, and particularly data available to anyone on the network without reference to the physical location of the resource or the user. Most companies have their networks designed to share a common printer, which at the long run is economical. Such activities like airline bookings, online shopping and other business can be done at your convenient place; you do not have to be physically present to do it.

In government institutions, computer networking can be used to store important data in a centralized location, which allows other computers in the network to retrieve data from the main location. A government can set up a good network, which enables employees to share resources like the printers. Employees can also share files by copying files from another computer. One can move from one meeting to another with a tablet and continue to work in the document from where it was left. ("Main benefits of computer networking in 2017," 2017)

Home networks are now used for entertainment including creating music, photos, and videos. Netflix and Soccer fans can now watch at the comfort of their houses by use of internet. Internet access connects home users to remote computers. As with companies, home users can access information, communicate with other people, and buy products and services with e-commerce (Tanenbaum & Wetherall, 2010).individuals are now able to use mobile computers, like laptop and handheld computers to communicate to each other (Tanenbaum & Wetherall, 2010). People on the go prefer to use their mobile devices to read and send email, tweet, watch movies, download music, play games, or simply to surf the Web for information (Tanenbaum & Wetherall, 2010).

2.2.2 Types of networks

Network types are classified according to scale. Distance is an important criteria in classification because technologies are used at different scales. In this chapter, four types of networks are described: Personal Area Networks (PAN), Local Area Network (LAN), Metropolitan Area Network (MAN) and Wide Area Network (WAN)

Personal Area Networks (PAN) let devices communicate over the range of an individual. A common example may be a wireless network that connects a computer with its peripherals (Tanenbaum & Wetherall, 2010). A diagrammatic illustration of PAN is shown in figure 2.1

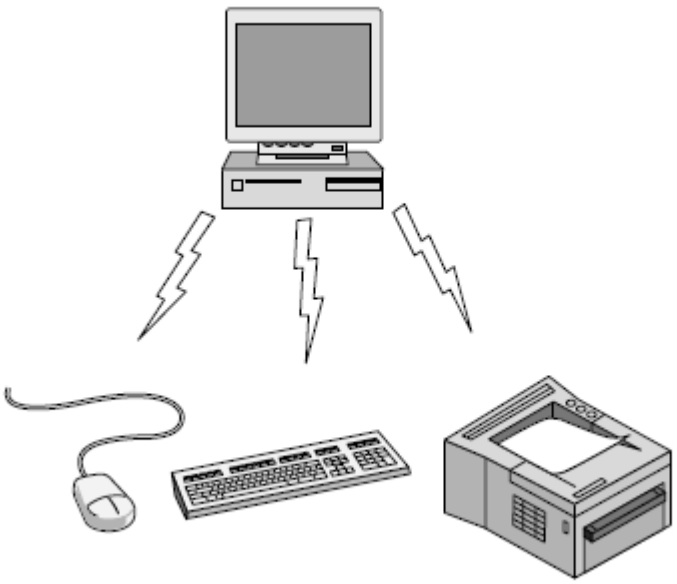


Figure 2.1: A Bluetooth PAN configurations (Tanenbaum & Wetherall, 2010)

A Local Area Network (LAN) can be a privately owned network that operates within and nearby one building sort of a home, office or factory. LANs are widely used to connect personal computers and consumer electronics to allow them to share resources (e.g., printers) and exchange information (Tanenbaum & Wetherall, 2010). A diagrammatic illustration of LAN is shown in figure 2.2

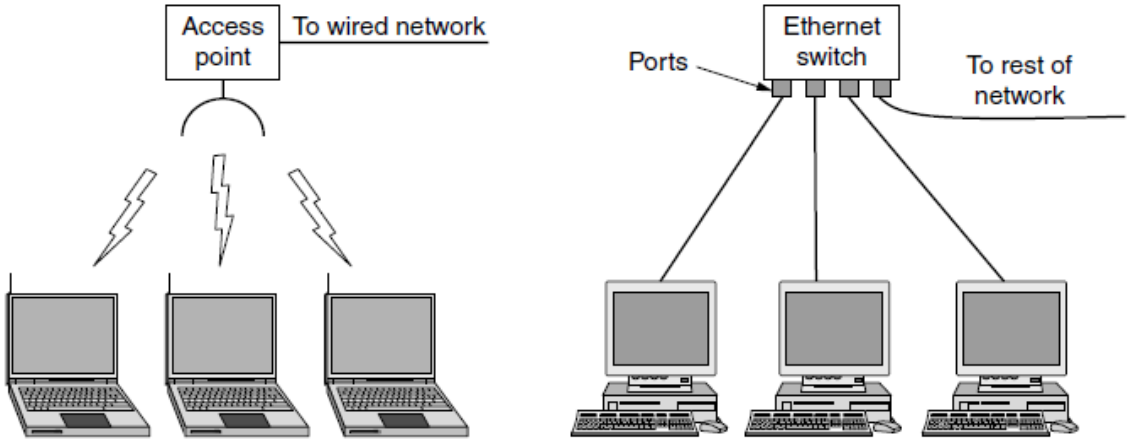


Figure 2.2: Wireless and wired LAN (Tanenbaum & Wetherall, 2010)

A Metropolitan Area Network (MAN) covers a city. The best-known samples of MANs are the cable television networks available in many cities. These systems grew from earlier community antenna systems utilized in areas with poor over-the-air television reception. In those early systems, an outsized antenna was placed on top of a close-by hill and a sign was then piped to the subscribers' houses (Tanenbaum & Wetherall, 2010). A diagrammatic illustration of MAN is shown in figure 2.3

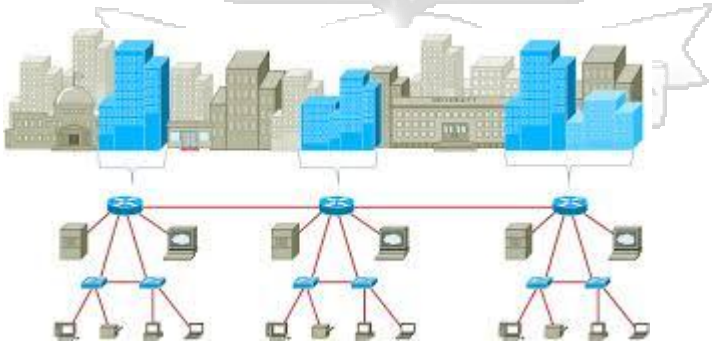


Figure 2.3: Metropolitan Area Network covering three cities (http://generalnote.com/)

A Wide Area Network (WAN) spans an outsized geographic area, often a rustic or continent. We will begin our discussion with wired WANs, using the instance of a corporation with branch offices in several cities (Tanenbaum & Wetherall, 2010)

A diagrammatic illustration of MAN is shown in figure 2.4

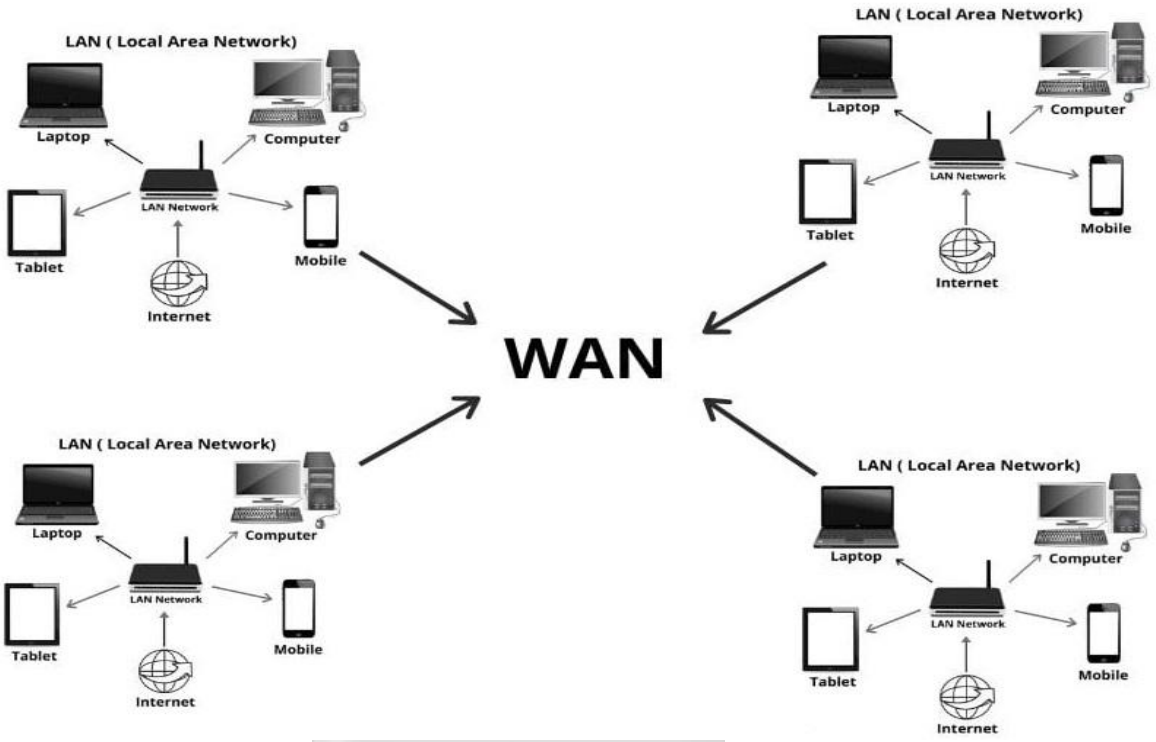


Figure 2.4: Wireless and wired LAN (digitalworld839.com)

2.3 Common threats, vulnerabilities and attacks for computer networks

Network security is where modification, disclosure or destruction of data is prevented by protecting the network, its applications and services from unauthorized access (Ahmad & Habib, 2017). Level of access is defined by security policies of a company, it describes who is permitted or denied access to a certain resource. Such security mechanisms are important since it helps deal with new attacks.

2.3.1 Network threats

A threat is a potential risk that takes advantage of a vulnerability to compromise security and cause potential disruption to service offered by computer systems (Ciza, 2020). A threat could either be intentional or unintentional. Human accidental errors and omissions are examples of threats which contribute in security problems and occur anywhere in the system. Errors could occur during data entry and cause a system to crush if a number of errors occur while other errors could occur during maintenance or installation, which also causes a security threat for security. Errors breach the integrity of data (Ahmad & Habib, 2017)

Increase in information technology increases the threat of fraud and theft. Attackers find new methods to attack a system, such frauds are those that involve monetary accounts. Attackers

not only target financial systems, other systems with resources or controls are also under threat such as university grading system, inventory system and human resource attendance system (Ahmad & Habib, 2017). Threat of fraud and theft can be from either an insider or outsider though the insiders do most frauds because they are authorized and understand vulnerabilities that exist within a system hence in a better position to conduct a crime. Previous employees of any organization whose accounts have not been deleted well and on time can be a threat for the company. Disgruntled employee is an employee who is dissatisfied with his/her job. Such an employee understands the flaws in a network and is well aware of the actions that can cause the most damage. (Weaver & Farwood, 2013)Downsizing in any organization can be an opportunity for those employees with enough information to attack the system. Common sample of sabotage are incorrect data modification, deletion of data, copying information and using it for the wrong purpose. Threats of this kind can be decreased by deleting accounts of disgruntled employees in a timely manner.

A person who tries to access computer systems with intentions of stealing or corrupting information is termed as a hacker. Hackers are the most dangerous threats in organizations and can be from inside or outside the organization (Ahmad & Habib, 2017). Organizations hire ethical hackers to seek out vulnerabilities in their network security mechanism. Such type of hacking is called ethical hacking and the hacker is called an ethical hacker (Ahmad & Habib, 2017)

2.3.2 Network vulnerabilities

System vulnerability is a weakness in the system due to its design or implementation. It includes security procedures and security controls associated with the system that could be exploited intentionally or unintentionally to cause damage or manipulate its confidentiality, integrity or availability of an information system (Dempsey, Nieles & Pillitteri, 2017)

Network vulnerability is anything that poses a possible avenue for attack or security breach against a system. (Awodele, Onuiri & Okolie, 2012). It includes viruses, passwords written on sticky pads, incorrectly configured systems amongst others. This sort of vices increase the danger to a system, however there is a wider context to the present concept than are stated above also as within the security community

In view of the foregoing and within the context valuable to security professionals, network vulnerability may be a security exposure that has the propensity to cause an unexpected event that compromises the security of a network infrastructure due to existence of a weakness, design, or implementation error. Network vulnerability is a flaw within a system that makes it impossible even where implementation and deployment is properly done, to prevent an intruder from illegal access to a network and data alteration. When the vulnerability is software oriented, such flaws should be fixed by the vendor through the release of patches (Awodele, Onuiri, & Okolie 2012)

Some network/system vulnerabilities include insecure ports, improper system configuration, poor anti-virus implementation, poor firewall deployment, poor intrusion detection system (IDS) setups, weak password implementation, and easy access to information, downloading of files and applications from unsecure websites, unsecure applications as a result of poor programming practices, application backdoors and lack of appropriate security policies.

2.3.3 Networks Attacks

An attack is any action that compromises security. It compromises the system by locking it down so that authorized persons are denied from using the system (Ahmad & Habib, 2017).Examples of Network attacks are Denial of Service (DoS), probing attack, Root to Local (R2L), Root to Local (R2L), and User To Root (U2R) amongst others.

Denial of Service (DoS).

This is an attack where legitimate users of an application or a network are prevented to use the resources of that service by overloading computing resources (Abraham, Das, Panda& Patra, 2011). It overloads resources by flooding the system with unnecessary information and consequently denies authorized users from accessing the system and denies legitimate users to make request or access the services (Niak, 2019).

Probing attack

This is a type of attack which collects of target systems before initiating an attack (Abraham, Das, Panda& Patra, 2011). Attackers will send attacks to know the topology of a network or identify machines with vulnerable software running on them (Niak, 2019). These attacks have become more sophisticated and have gone from just open scans that a firewall can stop, to

stealthy scans that obfuscate the attack more and may sneak through a firewall, revealing information a few machine's open ports and vulnerable software Practical Automated Detection of Stealthy Portscans (2002).

Root to Local (R2L)

Root to Local attacks happens when an attacker who does not have an account on a remote machine sends a packet over a network, exploits some vulnerabilities and gain local access as a user of the machine (Abraham, Das, Panda & Patra, 2011).

This is usually done after a scanning attack. After scanning a machine for open ports, an attacker can send packets to those ports to undertake to get local access. This can happen in some ways, like guessing the password of the machine, exploiting the FTP protocol, or employing a backdoor to realize access to the machine (Niak, 2019).

User To Root (U2R)

This is where an attacker exploits the systems vulnerability by gaining root access to the system yet he only has permission to access as a normal user (Abraham, Das, Panda & Patra, 2011). An unprivileged user can exploit a bit of vulnerable software to overwrite its stack with shell code, which results in the program opening a root shell for the attacker. This form of privilege escalation can help the attacker get access to sensitive information and permit them to wreak havoc on the machine or on the greater network.

Other attacks are Man in the Middle attack (MITM) viruses and malicious programs. MITM attack is an attack that happens when a hacker intrudes in between two communication parties and intercepts data or modifies the data (Ahmad & Habib, 2017).

2.4 Techniques for detecting intrusions

IT insecurity has become a serious concern due to increased crimes; anyone connecting their system to the internet is exposed to computer threats and attacks. Therefore, it is important to understand ways to secure systems from such crimes. Existing techniques that help detect and prevent such security crimes are Firewalls, honey pots, IPS and IDS.

2.4.1 Firewall

A firewall is a combination of hardware and software that isolates an organizations internal network from the internet allowing some packets to pass and blocking others (Kurose & Ross, 2013). A firewall blocks incoming data that is likely to contain attacks from a hacker. Information about the network is hidden by making look like the outgoing traffic is originating from the firewall instead of the network.

Packet filtering

Packet filtering also called stateless firewall is the most common type of a firewall.ech packet is treated individually and the filter looks at each packet entering or leaving the network and based on the rules it can accept, forward the packet to its destination, reject or drop the packets. (Kurose & Ross, 2013)

Stateful filters

This type of filter maintains records of all connections passing through it and is able to determine if it is part of a new connection, existing connection or it could be an invalid connection (Flanigan, 2018).The firewall does this by keeping an entry in a cache for each open flow. When the firewall encounters a packet it matches it against the rule.

Application layer

This type of firewall protects the network resources by filtering messages at the application layer. It understands the details of the application layer such as the application protocol and data, and intercepts any information intended for that application. The information is then used to determine if the data is a threat to the network security system that is meant to protect the resources of a network (Krit, 2017)

2.4.2 Honey pots

Honey pots are computer systems that are set up in a network to monitor trail of hackers (Flanigan, 2018). They are designed to attract hackers using the idea that a monitoring system will allow the attacker to be observed and then the system alerts the network administrator in case of an intrusion. Honey ports log attempts, monitor keystrokes of attackers which give

early warning signs (Banuprasad & Yang, 2005).It is a server-installed application specifically designed to monitor potential activities intended for an attack and observe intruders on how they get into the server of a computer system (Thu, 2013). Various types of honey pots are in existence based on the level and extent of activity. These types include;

Physical Honey pots

These are the types considered as physical tests which are functional and monitored heavily. It could be physically unsecured but it does not mean that permission has been granted for anyone to take it (Flanigan, 2018)

Virtual Honey pots

This is a host set up as a sacrificial host; it is set up on a network which has services running on a real operating system but contains fictional information (Flanigan, 2018). This type of honey pot will appear as a rogue infrastructure and cause internal stalling should it not be formalized in configuration, release and without change of management processes.

Low Interaction

These types of honey pot are limited in their degree of interaction. It appears to an attacker as an access point. It doesn't keep important data and therefore not much system requirements is required .It only logs probing activity and considers all attempts as suspicious (Malanik & Kouril, 2013)

2.4.3 Intrusion detection and prevention systems (IDS & IPS)

An IPS is a device that filters out suspicious traffic (Baykara & Das,2018).An IDS is a device that observes traffic that could be malicious and then generates alerts when it finds a potential intrusion(Kurose&Rosss,2013).More details of IDS are discussed in the following section

2.5 Intrusion Detection Systems (IDS)

An IDS is a device, software or a combination of both which monitors Network and its system activities. When monitoring, an IDS looks for malicious traffic, when its sees a potential intrusion it generates an alert. (Kurose & Ross 2013).An IDS 'performs deep packet

inspection,' which not only examines the header fields but also payloads in the datagram which includes the application-layer data(Kurose & Ross 2013).It contains a database of packet signatures that are part of attacks which automatically updates when new attacks are discovered. When packets pass through the IDS, header fields and payloads are matched to the signatures contained in a database. If a match is found an alert is created (Kurose & Ross 2013).

A diagram on Components of IDS is illustrated in figure 2.5

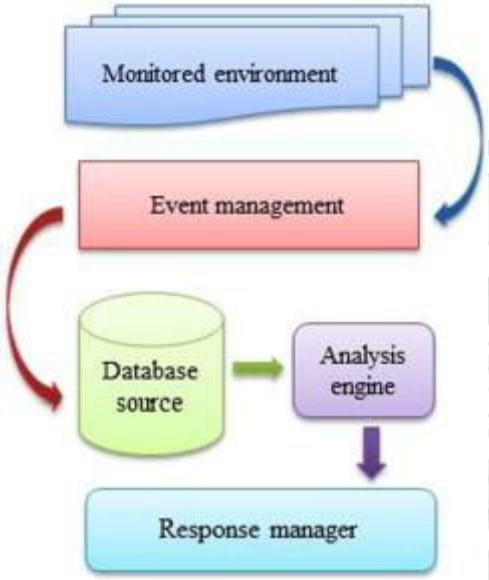


Figure 2.5: Functional components of IDS (Bace, 2000)

The first component of an intrusion detection system is a data source or otherwise known as event generator. Data source can be characterized as host based or network based monitors. The second component of an intrusion detection system the analysis engine, it takes information from the data source and looks for signs of attacks or deviations from the normal. The analysis engine can use Anomaly or Signature-Based Detection approaches. The third component is the response manager, which acts by informing through an alert when there is any sign of intrusion attack in the system.

A diagram on how an IDS works is illustrated in figure 2.6

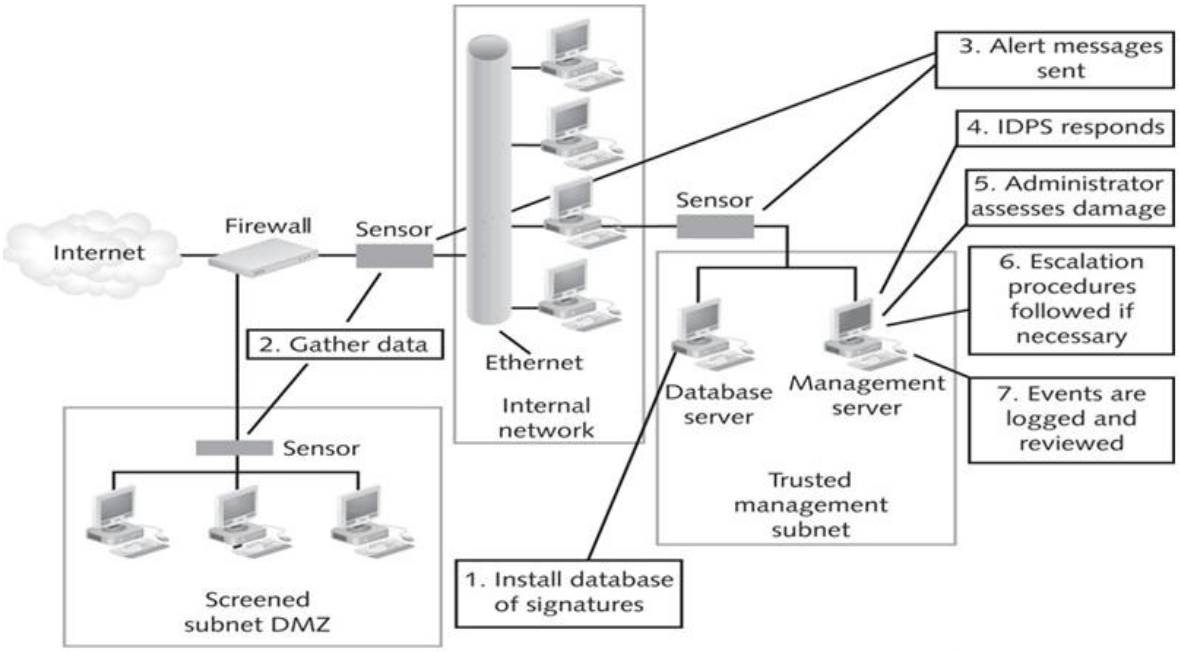


Figure: 2.6: Steps in intrusion detection, adopted from *Guide to Network Defense and Countermeasures, Third Edition,* (2013).

An IDS contains sensors placed in every section whose primary function is to research traffic and respond when attacks are detected. (Tanenbaum & Wetherall, 2010) The sensor examines each and every packet's header and data content that enters the network. The sensor looks for a pattern and behaviour within the network traffic that indicates malicious activity and sends alerts to the management console. The sensor examines the packets and checks against the user-defined policies or rule sets, which contains the priorities of the attacks to be monitored and therefore the counter measures to be taken when an attack is detected. If an attack is detected, the sensor sends an aware of the management console, logs the alert, and responds to the attack as per the defined policy (Tanenbaum & Wetherall, 2010)

Policies for sensors can be configured to generating alerts, log events, reset TCP connections, block firewalls traffic, scrub malicious packets, and also drop the packets before reaching the intended destination.

2.6 Classification of Intrusion Detection Systems

IDS are categorized based on two criteria, place of deployment of the IDS and detection technique used.

2.6.1 Classification based on the place of deployment (location)

IDS can be deployed at different location based on the results one wants to achieve. For example, Network based IDS monitors events at the network level while host based IDS monitor events at the host level (Toth, 2003)

The two types are Network-based Intrusion Detection System (NIDS) and Host-based Intrusion Detection System (HIDS)

NIDS-Network Intrusion Detection System

NIDS-is a system that detects network attacks as the payload is analyzed (Demertzis & Iliadis, 2014).It monitors traffic that travels through a network by use of sensors at various points to check packets entering a network, if there is any anomaly it either raises an alarm, creates a log or respond based on how the systems is designed(Kumaravel,2016)These type of systems have been considered successful and is widely used in the market (Shin, Kwon, Jo, Park, & Rhy, 2010).

NIDS inspects the network traffic in detail before permitting the traffic to pass through it in order to detect existence of an attack. Due to this, such systems are required to operate with minimum latency to achieve seamless network traffic speed (Kumaravel, 2016). However, this is challenging because of the rapid increase in the volume of network traffic. A diagrammatic illustration of a Network based IDS is shown in figure 2.7

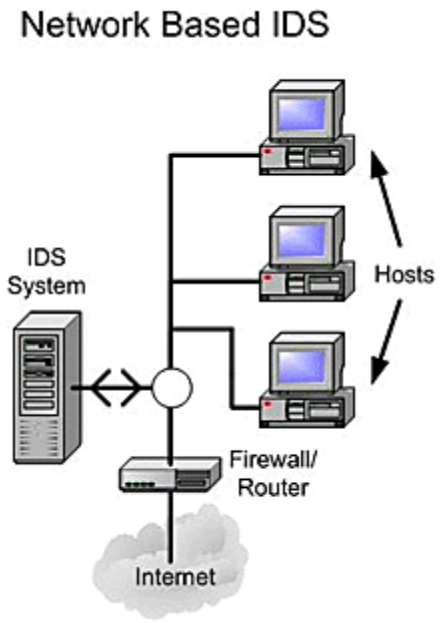


Figure 2.7: Architecture of Network based IDS as adopted from *Bauman National Library*

HIDS-Host Based Intrusion Detection System

HIDS-is a system installed on a host machine to control the software and investigate activities taking place on the system (Sarmin & Vasumathi, 2017).It monitors access to the file system, verify chains of system calls, or malicious changes then alerts on suspicious system activities. An example of host-level IDS is an anti-virus program running on a local machine (Kumaravel, 2016) one disadvantage of host-based IDS is security for the host. In case an attacker compromise the IDS then he can take complete control of the host (Berthier, Sanders, & Khurana, 2010)

A diagrammatic illustration of a Network based IDS is shown in figure 2.8

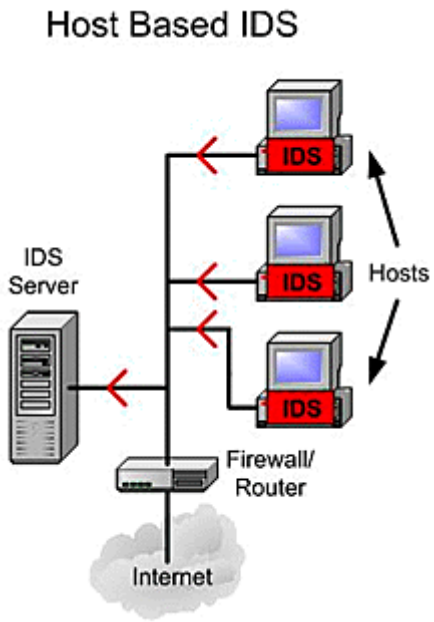


Figure 2.8: Architecture of Host based IDS (Sarmin & Vasumathi 2017)

2.6.2 Classification of IDS based on detection technique

For successful detection of attacks by IDS, systems should have an effective detection mechanism with high detection rates. Such detection mechanisms are common to the host-based IDS and network-based IDS (Kumaravel, 2016). Detection techniques can be classified as Signature/misuse based or Anomaly based (Kurose & Ross 2013).

An illustration of an intrusion detection system Architecture is shown in figure 2.9

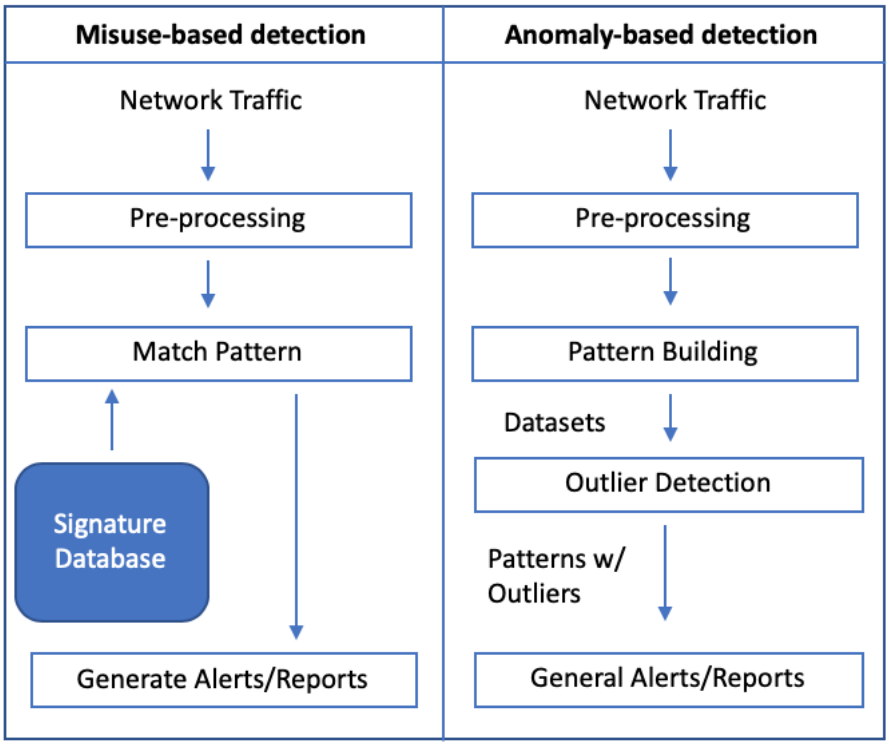


Figure 2.9: Architecture of an intrusion detection system (Mishra, Pilli, Varadharajan & Tupakula, 2016)

Signature Based IDS

A signature oriented IDS holds a database of attack signatures where a signature is a set of rules related to an intrusion activity. Network connections have signatures such as metadata and file fingerprints (MD5 or SHA1 hash) which can be used to determine whether a connection is normal or a potential attack (Kurose & Ross 2013). This is done by comparing the signature of an incoming connection to the signatures of previous attacks that are stored in a database (Ambedkar, Ambedkar & Raw, 2016). The problem with this approach is that if the database is not updated, the system will fail to identify new attacks.

Anomaly Based IDS

In anomaly-based detection, previous signature of the attack is not stored in the database. The system establishes a baseline for normal behaviour and constantly searches through Networks for abnormal behaviour. If it suspects any deviant Network behaviour, it classifies it as an anomaly (Alyasiri, 2018). The challenge with this approach is that it has a high propensity for

frequent false positives because some of the detected changes to the system are legitimate. To address the challenges of these traditional methods, Machine Learning techniques have been proposed for intrusion detection in the recent past (Wang, 2009)

2.6.3 By Evolutionary approach (traditional & modern)

Packet filters which are termed as traditional and stateful were used to inspect IP, TCP, UDP, and ICMP header fields when deciding which packets to let pass through the firewall. (Kurose & Ross, 2013)
Due to many attack types that need to be detected, deep packet inspection is required to be done in order to look beyond the header fields and into the actual application data that the packets carry.
Therefore there is a niche for another device that not only examines the headers of all packets passing through it (like a packet filter), but also performs deep packet(Kurose & Ross,2013) when intrusion detection systems were first defined in 1990s ,revenue was increasing at that time (Pirc, 2017).Network intrusion detection systems such as SNORT and Bro use hand written rules to detect signatures of known attacks, like a selected string within the application payload, or suspicious behaviour, like server requests to unused ports.
Many IDS systems available are mostly signature-based detectors. As much as they are effective at detecting known intrusion attempts and exploits, they fail to recognize new attacks and a carefully crafted variant of old exploits (Stolfo & Wang, 2004). A new generation of systems on machine learning is now being considered for anomaly detection. Anomaly Detection systems scheme expected behaviour in a system, and detect deviations of interest that indicate an attempted attack (Stolfo & Wang, 2004)

2.7 Anomaly based IDS

Anomaly based IDSs scheme expected behaviour in a system. It detects deviations of interest, which could indicate an attempted attack (Wang & Stolfo, 2004)

Anomaly detection systems like SPADE, ADAM, and NIDES learn a statistical scheme of normal network traffic; generate alarm when there are deviations from this scheme. Schemes are usually supported by the distribution of source and destination addresses and ports per transaction (TCP connections, and sometimes UDP and ICMP packets) (Mahoney2003)

IDSs uses network traffic baselines to work out a “normal” state for the network and compare current traffic thereto baseline. If deviation in network behaviour is detected, IDS will assume an attack has occurred and alerts the administrator.

The main advantage of anomaly based ID systems is that they don’t rely on previous knowledge about existing attacks; they can detect new attacks that have never been documented. However, it suffers from high warning rates which are created by inaccurate profiles of “normal” network operations and consequently compromise the effectiveness of the IDS (Zhang & Zulkernine)

Anomaly based IDS creates traffic profile as it observes traffic in normal operation, it updates normal profiles dynamically from the changes in the network traffic and integrate new profiles identified (Bhuyan, Bhattacharyya & Kalita,2011) it then looks for packet streams that are statistically unusual such as extreme growth in port scans and classifies them as anomalies (Kurose & Ross 2013).

The main advantage of anomaly based IDS is that they don’t rely on previous knowledge about existing attacks, they can detect new attacks that have never been documented.

2.7.1 Approaches to Anomaly Detection Systems

In anomaly intrusion detection, intrusion deviations from the normal behaviour can be identified using approaches such as predictive and statistical mechanisms. Statistical approaches use previous behaviour as a baseline to determine an intrusion deviation while predictive approach uses a rule base of user profile to predict future intrusions (Chen& Teng, 1990). The normal usage patterns are constructed from the statistical measures of the system features, for example, the CPU and I/O activities by a particular user or program.

Statistical approaches

Statistical approaches use previous profiles of an intrusion as a baseline to compare with the recent behaviour of an intrusion, for example it can compare the behaviour of a user of a computer system with observed behaviour to identify any deviation. If a deviation is identified, it is considered as intrusion. This approach requires a construction of a scheme to be used for normal user behaviour so that if there is any user behaviour that is unusual it is flagged as an intrusion. IDSs have previously used statistical approach to detect intrusions (Lunt, 1993).

Profiles are frequently updated to allow the system to learn new behaviour. This approach is advantageous as it learns the behaviour of users automatically and thus more sensitive than humans however, this approach is quite risky as it can be trained by intruders to consider intrusive events as normal hence make the intruders undetected (Abraham, Grosan, Peddabachigari & Thomas, 2007)

Predictive pattern generation

Predictive approach generates a pattern using a rule base of user profiles as a guide to predict future events based on previous occurrences (Chen & Teng, 1990)

An example of a rule can be like this one below

E1 — E2 --> (E3 = 80%, E4 =15%, E5 = 5%)

E's are events from security audit trail and the percentages represent the probability of an event occurring. It means that the observed events in sequence E1, E2 and E3 has the probability of event E3 occurring is 84%, E4 as 15% and E5 as 5%. (Abraham, Grosan, Peddabachigari & Thomas, 2007).If the sequence of observed events matches the rule on the left hand side but the following events significantly deviate from rule on the right hand side then an intrusion is said to have occurred (Abraham, Grosan, Peddabachigari & Thomas, 2007).This approach has the ability to detect and respond quickly to anomalies, easier to detect users who attempt to train the system during its training period .However it cannot detect some intrusions whose sequence of events have not been recognized and created into the rules.

2.8 Machine Learning Techniques for Intrusion Detection

Machine learning advances' made in the recent past have proven to be efficient in applications including Intrusion Detection Systems (Salman, Bhamare, Erbad, Jain, & Samaka, 2017)they are trained comprehensively to improve the efficiency of anomaly detection with less complexity(Tsai, Hsu, Lin, & Lin, 2009) In machine learning, the behaviour of a system is automatically learned (Wang & Stolfo, 2004)

Various machine technologies used for intrusion detection are discussed in this chapter.

2.8.1 Support Vector Machine (SVM)

This is a supervised learning method for classification. In SVM, a hyper plane is constructed to separate various classes of instances in high-dimension space. Finding the hyper plane is an optimization problem, which maximizes the distance between the hyper plane and the nearest data point in different classes. SVM is employed to detect failures and compared with other methods, Similar to Logistic Regression and Decision Tree, the training instances are event count vectors together with their labels. (Cao, Liang, Li & Xu, 2016) In anomaly detection via SVM, if a new instance is located above the hyper plane, it would be reported as an anomaly, while marked as normal.SVM has superior properties of fast training, scalability and generalization capability give them an advantage in the intrusion detection application however slow in multi class identification (Janoski, Mukkamala & Sung, 2002).From the experiment performed by Hu (2003) on the noisy data, the SVM performed very poorly, it obtained 60% true positives at 10% false positives.

2.8.2 Decision Tree

Decision Tree may be a tree structure diagram that uses branches for instance the anticipated state for every instance. The decision tree is built in a top-down way with use of training data. Every tree node is formed using the current "best" attribute, which is chosen by attribute's information gain (Han, Kamber & Pei, 2011). For example, the root node in the figure below illustrates that there are totally 20 instances in our dataset. An example of Decision Tree is illustrated in figure 2.10

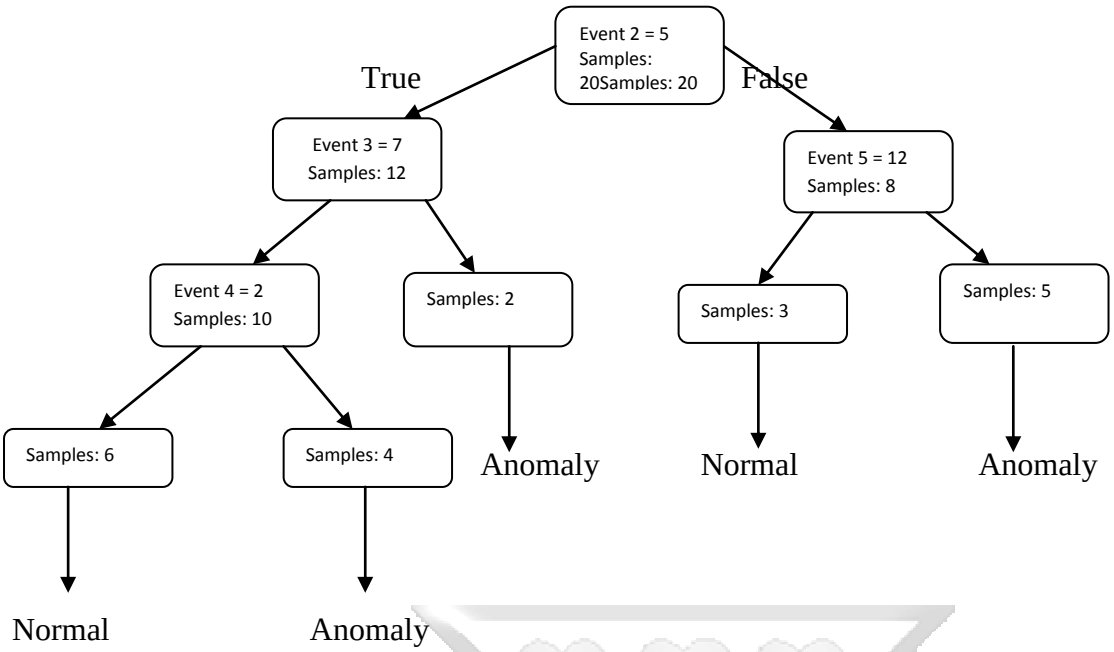


Figure 2.10: An example of a decision tree as adopted from He, Lyu and Zhu (2016)

When splitting the root node, the occurrence number of Event 2 is treated as the "best" attribute. Therefore, the whole 20 training instances are divided into two subsets according to the value of this attribute, in which one contains 12 instances and the other comprises of 8 instances. The event count vectors together with their labels are utilized to build the decision tree. To identify the state of a new instance, it criss-crosses the decision tree according to the predicates of each traversed tree node. In the end of crisscrossing, the instance will arrive at one of the leaves, which reflects the state of this instance. As much as this method obtains more accuracy compared to other schemes, Sabhnani and Serpen (2004), it demonstrates that this method is very sensitive to training (Gharibian & Ghorbani, 2007).DTs suffers from the drawback of not being able to deal well with unseen data. New attacks may be classified as some default class, such as 'normal'

This classifies new/unseen data as a new 'unknown' class. By doing this, they avoid a significant amount of misclassifications of new attacks as normal connections, (Kobayashi, Ohta & Kurebayashi, 2008)

2.8.3 Neural Networks

With the aim of having a simulation of how the human brain operates (featuring existence of neurons and of synapses among them), neural Networks have been deployed in the field of

anomaly intrusion detection, majorly due to their flexibility and adaptability to environmental changes. This detection approach has been employed to create user profiles Fox, Henning, Reed and Simonian, (1990) to predict the next command from a sequence of previous ones Becker, Debar and Siboni, (1992) to identify the intrusive behaviour of traffic patterns (Bonifacio, Cansian, Carvalho & Moreira, 1998). However, a similar feature in the proposed variants, from recurring neural Networks to self-organizing maps Ostermann, Ramadas and Tjaden (2003) is that they do not avail a descriptive scheme that tells why a particular detection decision has been considered. Neural networks advantage is that they cope with noisy data, and is easy to modify. The disadvantage with this approach is that it can be trained by an intruder during its learning phase.

2.8.4 Bayesian Networks

A Bayesian Network is a scheme that translates probabilistic relationships amid variables of choice. This method is commonly used for intrusion detection in combination with statistical schemes, a procedure that yields several merits (Heckerman, 1995), including the capability of encoding interdependencies between variables and of event predictions, as well as the ability to incorporate both prior knowledge and data, faster in learning and classification tasks and is easy to update when there are new cases.

In this technique, unnecessary communication and processing overload are prevented since the events used to estimate the probability of the attacks are inspected at the location of the network where it occurred. Hence, the problem of various control record mismatch does not arise (Moorthy & Sathiyabama, 2012). It provides us with the estimate of the probability that an attack is going on when the network is fed with the needed data (Mehdi, Anou, Bensebti, Zair & 2007). However, the disadvantage of using Bayesian Networks is that it requires higher computational effort.

2.9 Existing solutions for IDS based on Machine Learning

Altwaijry & Algarny (2011) performed a number of experiments on Bayesian networks using 10% of KDD dataset. Five sub experiments were conducted using the same training engine for normal records and attack records.

In the first experiment normal records were detected with a high accuracy = 99.03%.the best detection rate of 99.36% was for DOS attack type and worst results was 0% for U2R and R2L attacks. The results of the experiment are tabulated in table 2.11

Table 2.11: Experiment 1 test results percentages (Altwaijry & Algarny, 2011)

Test	TN	TP	FN	FP	DR	CR
All attacks	99.03	89.70	0.97	10.30	89.70	94.37
DOS	99.03	99.36	0.97	0.64	99.36	99.20
Probing	99.03	57.15	0.97	42.85	57.15	78.09
U2R	99.03	0.00	0.97	100.00	0.00	49.52
R2L	99.03	0.00	0.97	100.00	0.00	49.52

In the second experiment an accuracy of 99.6% TN was detected for the normal records with the best DR being for DOS attack type at 99.24%.Worst result of 0% was noted for U2R and R2L attacks. The results of the experiment are tabulated in table 2.12

Table 2.12: Experiment 2 test results percentages (Altwaijry & Algarny, 2011)

Test	TN	TP	FN	FP	DR	CR
All attacks	99.60	65.50	0.40	34.50	65.50	82.55
DOS	99.60	99.24	0.40	0.76	99.24	99.42
Probing	99.60	17.73	0.40	82.27	17.73	58.67
U2R	99.60	0.00	0.40	100.00	0.00	49.80
R2L	99.60	0.00	0.40	100.00	0.00	49.80

In the third experiment, normal records are detected with high accuracy of 99.4%, the best accuracy being 91.5% for U2R attack type. The results of the experiment are tabulated in table 2.13

Table 2.13: Experiment 3 test results percentages (Altwaijry & Algarny, 2011)

Test	TN	TP	FN	FP	DR	CR
Not normal	99.40	71.00	0.60	29.00	71.00	85.20
DOS	99.40	70.30	0.60	29.70	70.30	84.85
Probing	99.40	81.90	0.60	18.10	81.90	90.65
U2R	99.40	91.50	0.60	8.50	91.50	95.45
R2L	99.40	61.50	0.60	38.50	61.50	80.45

The fourth experiment shows high accuracy of TN 99.7% for the normal records, with best DR being for the U2R attack type which was 93.0% and the worst result at 0.02% for DOS attacks. The results of the experiment are tabulated in Table 2.14

Table 2.14: Experiment 4 test results percentages (Altwaijry & Algarny, 2011)

Test	TN	TP	FN	FP	DR	CR
Not normal	99.70	76.50	0.30	23.50	76.50	88.10
DOS	99.70	0.02	0.30	99.98	0.02	49.86
Probing	99.70	71.60	0.30	28.40	71.60	85.65
U2R	99.70	93.00	0.30	7.00	93.00	96.35
R2L	99.70	61.50	0.30	38.50	61.50	80.60

In the fifth experiment, accuracy for normal records decreased to 68.03%, the best DR being 96.3% for U2R attack type and the worst result at 25% for DOS attacks type. The results of the experiment are tabulated in table 2.15

Table 2.15: Experiment 5 test results percentages (Altwaijry & Algarny, 2011)

Test	TN	TP	FN	FP	DR	CR
Not normal	68.03	10.00	31.97	90.00	10.00	39.02
DOS	68.03	2.00	31.97	98.00	2.00	35.02
Probing	68.03	63.60	31.97	36.40	63.60	65.82
U2R	68.03	96.30	31.97	3.70	96.30	82.17
R2L	68.03	85.35	31.97	14.65	85.35	76.69

From the experiment it can be concluded that Bayesian networks give a high detection accuracy though its accuracy can still be improved by using several Bayesian filters in parallel with each filter optimized to detect one type of record (Altwaijry & Algarny,2011).A major drawback for this classifier is its sensitivity while using some features .

Chen, Wang & Yang (2013) In their experiment used 10% of KDD 99 dataset to conduct simulation on a Naïve Bayes classifier, the characteristics of the data set which contains one type of normal data and other types of attacks which falls into one of the four attack categories (DoS, probe, R2L and U2R).The characteristics are as shown in table 2.16

Table 2.16: Characteristics of the 10% KDD '99 dataset (Chen, Wang & Yang 2013)

Class	Training samples	Testing samples	10% KDD training data distributions (%)	10% KDD test data distributions (%)
Normal	97277	60592	19.69	19.48
DOS	391458	237594	79.24	73.91
R2L	1126	8606	0.23	5.20
U2R	52	70	0.01	0.07
Probe	4107	4166	0.83	1.34
Total	494020	311028	100.00	100.00

In this experiment,simulation results shows how naïve bayesian outperfoms other classifiers in all attack types (DoS, probe, R2L and U2R) and normal attacks.

The advantage of Naïve bayes is its srong independence assumption, probability of an attribute does not affect the probability of another(Al-Sharafat & Naoum ,2009). However, Naïve Bayes classifier generates more false positives (Panda & Patra, 2007). This problem can be alleviated by reducing the false positives through classification using Bayesian network. Comparison of the overall classifier performance is shown in table 2.17

Table 2.17: Comparison of the overall classifier performance (Chen, Wang & Yang 2013)

Model	Normal	DOS	R2L	U2R	Probe
Proposed Naïve Bayesian	99.56	99.85	99.36	95.91	98.34
Decision tree	98.75	98.68	97.73	94.83	96.89
Support vector machine	99.23	98.49	96.88	95.57	93.71
Naïve Bayes	97.94	98.63	99.54	92.26	91.82
Artificial neural network	97.74	98.66	98.41	93.37	92.97
Genetic algorithm	97.69	98.92	98.72	93.18	90.99

KDD CUP 1999 data set was used for the experiment; the experiments were performed by selecting 18,285 records randomly similar to a prior research (Beghdad, 2008)

As shown in table 2.18 the details of the records and types of attack categories were captured.

Table 2.18: Number and distribution of training and test dataset (Hao, Huang, Ma & Wang, 2010)

Number and distribution of training and test dataset.

Connection type	Training dataset		Testing dataset	
Normal	3000	16.41%	60,593	19.48%
DoS	10,000	54.69%	229,853	73.89%
PRB	4107	22.46%	4166	1.34%
R2L	1126	6.16%	16,189	5.2%
U2R	52	0.28%	288	0.09%

Results of decision tree, Naïve Bayes, Back propagation Neural Networks (BPNN) and Fuzzy Clustering Artificial Network (FCANN) were compared using Weka tool (Witten & Frank, 2005). Five attack categories were used to compare the performance of the algorithms and the results are as shown from table 2.19

Table 2.19: Number and distribution of training and test dataset (Hao, Huang, Ma & Wang, 2010)

Performance comparison of various methods (normal).

	Decision tree	Naïve Bayes	BPNN	FC-ANN
Precision (%)	91.22	89.22	89.75	91.32
Recall (%)	99.41	97.70	98.20	99.08
F-value (%)	95.14	93.27	93.79	95.04

Table 2.20 shows the performance comparison of various methods (DoS)

Table 2.20: Performance comparison of various methods (Hao, Huang, Ma & Wang, 2010)

Performance comparison of various methods (DoS).

	Decision tree	Naïve Bayes	BPNN	FC-ANN
Precision (%)	99.84	99.69	99.79	99.91
Recall (%)	97.24	96.65	97.20	96.70
F-value (%)	98.52	98.15	98.48	98.28

Table 2.21 shows the performance comparison of various methods (PRB)

Table 2.21: Performance comparison of various methods (PRB) (Hao, Huang, Ma & Wang, 2010)

Performance comparison of various methods (PRB).

	Decision tree	Naïve Bayes	BPNN	FC-ANN
Precision (%)	50.00	52.61	60.94	48.12
Recall (%)	78.13	88.13	88.75	80.00
F-value (%)	60.98	65.89	72.26	60.09

Table 2.22 shows the performance comparison of various methods (R2L)

Table 2.22: Performance comparison of various methods (R2L) (Hao, Huang, Ma & Wang, 2010)

Performance comparison of various methods (R2L).

	Decision tree	Naïve Bayes	BPNN	FC-ANN
Precision (%)	33.33	46.15	57.14	93.18
Recall (%)	1.43	8.57	5.71	58.57
F-value (%)	2.74	14.58	10.39	71.93

Table 2.23 shows the performance comparison of various methods (U2L)

Table 2.23: Performance comparison of various methods (U2L) (Hao, Huang, Ma & Wang, 2010)

Performance comparison of various methods (U2L).

	Decision tree	Naïve Bayes	BPNN	FC-ANN
Precision (%)	50.00	25.00	50.00	83.33
Recall (%)	15.38	7.69	23.08	76.92
F-value (%)	23.53	11.76	31.58	80.00

From the tables we can see the difference of the evaluations under different attack categories i.e. normal, DoS, Probe, R2L and U2R. FC-ANN gets the highest precision, recall and F-value in most attack categories (DoS, R2L and U2R) than decision tree, Naïve Bayes and BPNN. However it performs poorly for PRB attacks which according to the table are among the low - frequent attacks i.e. R2L and U2R. Table 2.24 shows the performance summary of the various algorithms.

Table 2.24: Performance comparison of various algorithms (Hao, Huang, Ma & Wang, 2010)

Average accuracy, percentage of training successfully, and training time of various methods.

	Decision tree	Na Bayes	BPNN	FC-ANN
Average accuracy (%)	96.75	96.11	96.65	96.71
Percentage of training successfully (%)	100	100	60	100
Training time (s)	2.68	1.93	1538.17	2125.4

For training time, FC-ANN algorithm is seen to take more training time compared to decision tree, Naïve Bayes and BPNN the training time are as 2.68, 1.93, 1538.17 and 2125.4 s, respectively. The reason FC-ANN is seen to take more time is attributed to clustering time and every ANN training time.

Abraham, Grosan, Peddabachigari and Thomas (2007) compared two algorithms, Decision trees and Support Vector Machine used for detecting intrusions in a network. Two experiments were performed for decision tree and SVM. In decision tree experiment, five classifiers were constructed and data was partitioned into normal and attack classes where attack is the collection of four classes (Probe, DOS, U2R, and R2L) five different classifiers. The results are as in table 2.25

Table 2.25: Performance of Decision Tree using five attack types (Abraham, Grosan, Peddabachigari & Thomas 2007)

	Training time (sec)	Testing time (sec)	Accuracy (%)
Normal	1.53	0.03	99.64
Probe	3.09	0.02	99.86
DOS	1.92	0.03	96.83
U2R	1.16	0.03	68
R2L	2.36	0.03	84.19

The table gives a summary of the test data. Training and testing times of the classifier is shown in seconds and accuracy in percentage for all the five classes. The same experiment is repeated for SVM and the results are as shown in table 2.26

Table 2.26: Performance of SVM using five attack types (Abraham, Grosan, Peddabachigari & Thomas 2007)

	Training time (sec)	Testing time (sec)	Accuracy (%)
Normal	5.02	0.13	99.64
Probe	1.33	0.13	98.57
DOS	19.24	2.11	99.78
U2R	3.05	0.95	40.00
R2L	2.02	0.13	34.00

Evaluation of the two algorithms is done by comparing the performance of the two in terms of accuracy, training time and testing time are evaluated by comparing two experiments as summarized in the table 2.27

Table 2.27: Comparisons of SVM and Decision Tree (Abraham, Grosan, Peddabachigari & Thomas, 2007)

Class	Decision Tree			SVM		
	Training time(s)	Testing time(s)	Accuracy (%)	Training time(s)	Testing time(s)	Accuracy (%)
Normal	1.53	0.03	99.64	5.02	0.13	99.64
Probe	3.09	0.02	99.86	1.33	0.13	98.57
DOS	1.92	0.03	96.83	19.24	2.11	99.78
U2R	1.16	0.03	68.00	3.05	0.95	40.00
R2L	2.36	0.03	84.19	2.02	0.13	34.00

The summary shows that decision tree performs better in terms of accuracy for Probe, R2L and U2R classes compared to SVM while SVM showed better accuracy for DOS class of attacks and gave same performance for normal class in both algorithms. As depicted from the table above, decision trees give a good less training time and testing time compared to SVM. Decision tree also has the capability of multi-class classification which SVM doesn't have. Multi-class classification is a very useful feature for intrusion detection schemes.

2.10 Summary of gaps on the existing solutions based on Machine Learning

In an experiment that was conducted using KDD dataset, The performance of decision trees and Support Vector Machine were compared, it was found that decision trees was superior to Support Vector Machine in accuracy and detection; but Support Vector Machine was better than Decision Trees in false alarm rate (Su-Yun, Wu & Yen 2009)

Since Neural Networks is easy to modify intruders can take advantage of training the classifier during its learning phase.

Bayesian Networks was unable to perform well when dealing with categories with less number of records (Khor, Ting & Amnuaisuk, 2020) .This was because the dataset needed to be divided for training, building and evaluation purposes.

Sabhnani & Serpen, (2004) in their experiment examined the performance of several machine learning techniques including Decision Trees. It obtained good accuracy, but does not perform as well as other techniques on some classes of intrusion, like U2R and R2L attacks, Decision Trees are very sensitive to the training data and do not learn well from imbalanced data (Gharibian & Ghorbani, 2007)

2.11 Conclusion

Bayesian based Networks are powerful tools for decision and reasoning under uncertainty, faster in learning and classification tasks hence an efficient method for predicting outcome with high interrelated variables. Due to its incremental feature, new cases are easily updated hence suitable for anomaly attacks since they are constantly changing.

Naïve Bayes is therefore, a suitable approach for this study. It forms classification sets through re-sampling data in a manner that provides the most abundant training data to every successive classifier. It is an iterative algorithm whose main idea is to train diverse learning algorithms for the same training set.

2.12 Conceptual framework

The concept diagram presented in figure 2.28 represents the proposed solution that this study aims to implement. Normal data (Intrusive and non intrusive) are used to train the classifier and then anomaly data (Intrusive and non-intrusive) are used to test the classifier.

Figure 2.28 represents the proposed solution that is to be implemented in this research.

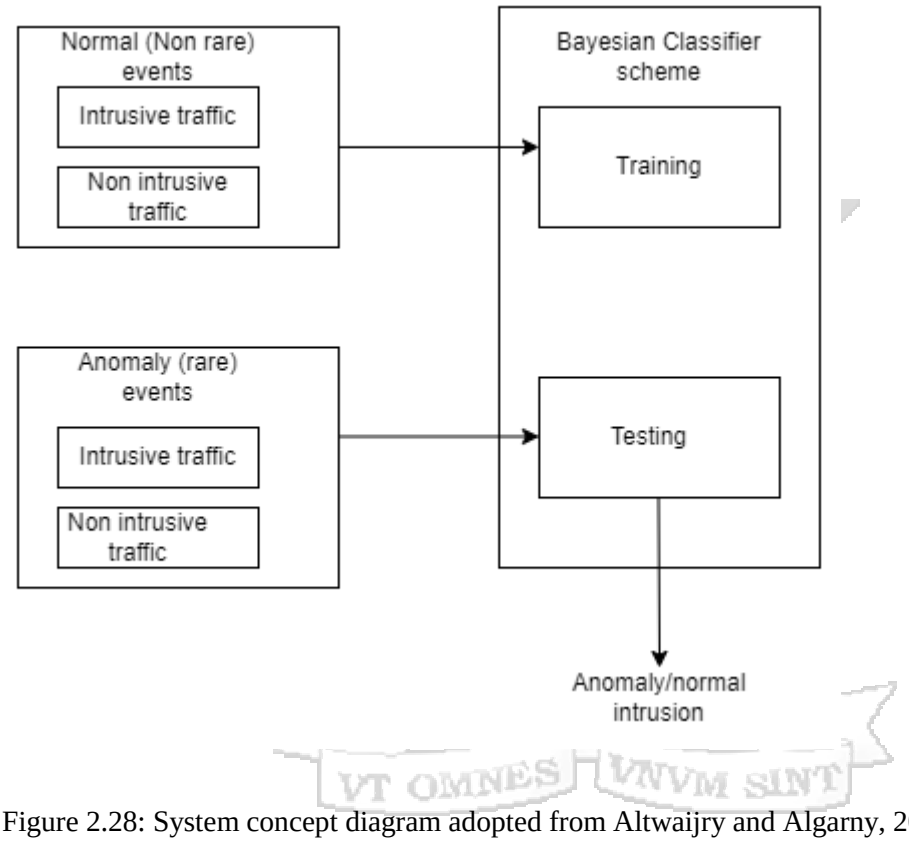


Figure 2.28: System concept diagram adopted from Altwaijry and Algarny, 2011

The classifier modelling consists of stages such as data collection, training and testing. In the first stage, data collected is manipulated and then trained. Normal samples are used to train the classifier so that it recognizes normal activity and then both anomaly and normal samples are then subjected to the testing detector where any sample that deviates from the trained detector is considered an anomaly.

CHAPTER 3: RESEARCH METHODOLOGY

3.1 Introduction

Methodology is a plan that looks at how the study through a systematic approach intends to answer the research questions and meet research objectives. This chapter explores literature review in order to understand the concept under research; which is used in answering research questions 1 and 2. Next step is the agile methodology, which guides in answering research questions 3 and 4.

3.2 Systematic literature review

The 1st and 2nd objectives were met by examining various existing literatures will gave insight into the issue of cyber-attacks and prediction that helped develop research hypothesis and delve into study exploration. The study was based on grounded theory and the dynamic Bayesian attack graph (DBAG), including other relevant schemes and theories to explain the concept under research (Rebai, 2010). The methodology covered an overview of a scoring scheme, its construction and implementation. The discovered theory helped in developing a conceptual framework that describes the cyber-attack and prediction phenomenon. Description made it possible to clearly and accurately describe the attributes of attack prediction situation and helped achieve in-depth understanding of threat prediction (Lapan, Riemer & Quartaroli, 2012).This is an approach that the research took to achieve its objective 1and 2.

3.2.1 Theoretical Basis of Study

Bayesian network concerns the inferences in discrete events of Bayesian networks which is a complete model for the variables and their relationships and can be used to answer probabilistic queries about them (Rebai, 2010). For instance, the network will determine updated knowledge of the state of a subset of variables when other variables (the evidence variables) are observed (Ding & Wang2010). This process of computing the posterior distribution of variables given evidence is called probabilistic inference. A Bayesian network can thus be considered a mechanism for automatically applying Bayes' theorem to complex problems. Any variable updating in any node of Bayesian networks might end in the evidence propagation across the Bayesian networks. How inferences are examined and executed is an important task within the application of Bayesian networks.

3.2.2 Probability Theory

The theory is based on the Bayes theorem, which is a statistical principle for combining prior knowledge of classes with new evidence gathered from data (Bayes, 1763). Every stage of situation assessment requires assigning prior probabilities to the hypotheses. These prior probabilities are obtained from knowledge of the prevailing situation. The most important capability of this theory is their ability to determine the probability that a certain hypothesis is true (e.g., a probability of an intrusion to be normal or abnormal) given a historical dataset. This art of prediction under uncertainty provides a framework for reasoning even when data is unseen. This framework is ideal for current cyber attack prediction since the attacks are constantly changing, a situation that lies at the heart of Bayesian probability theory.

The probability of a variable can occur if the value of another random variable is known. This is how the probability can be obtained. Let *A* and *C* be a pair of random variables. Their joint probability$P(A=a,C=c)$ refers to the probability that variable *A* will take on the value *a*, and a variable *C* will take on value *c*. The joint conditional probabilities for *A* and *C* are related as follows: $P(A,C)=P(C|A)\times P(A)=P(A|C)\times P(C)$

Rearranging the last 2 expressions, we get: $P(C|A)=P(A|C)P(C)P(A)$ P(C|A) is the posterior probability of C conditioned on A.

P(C) is the prior probability of C.

P (A|C) is the posterior probability of A conditioned on C.

3.2.3 Bayesian Networks

A Bayesian network is a probabilistic graphical model that represents the variables and the relationships between them .The network is constructed with nodes as the discrete or continuous random variables and directed edges as the relationships between them, establishing a directed acyclic graph (DAG). The child nodes are dependent on their parents. Each node maintains the states of the random variable and the conditional probability form. Bayesian networks are built using expert knowledge or using efficient algorithms that perform inference.

Bayesian network can be learned from domain experts, data or a combination of the two. The structure of the Bayesian network and the probability tables are provided by an expert based on her/his experience. A combination of a directed acyclic graph of nodes and links, and a set of conditional probability tables can describe uncertainty in cyber security using probability theory. Nodes can represent features or classes, while links between nodes represent the relationship between them. Conditional probability tables determine the strength of the links. There is one probability table for each node (attribute) that defines the probability distribution for the node given its parent nodes. If a node has no parents the probability distribution is unconditional. If a node has one or more parents the probability distribution is a conditional distribution where the probability of each feature value depends on the values of the parent.

The probability tables can be calculated from the available training data. Inferring unobserved variables, parameter learning, and structure learning are among the main tasks for training Bayesian networks.

3.3 Agile methodology

Agile methodology was used as a road map to achieve the results. It is a suitable approach since Machine Learning implementation goes through an iterative cycle each step of the entire process was revisited until quality data was achieved that was sufficient to train the scheme, otherwise the scheme could not be validated if not trained. The research was based on secondary data. KDD99 Dataset containing the Network traffic logs was downloaded from the UCI Machine Learning Repository for analysis in Weka tool. Two experiments were conducted using Weka machine learning tool and Python code in JupyterLab to train and test the classifier. The results for each approach was recorded independently and then compared. An analysis of the results from the experiment was done to evaluate the performance of the prototype in both platforms.

Two approaches were used to implement the experiment. Python and Weka framework. Python compared to Weka provides better performance in terms of correct and incorrect instances, precision and recall. However since Weka has inbuilt algorithms such as Support Vector Machine and other ML algorithms, it easier to spot check the performance of other machine learning algorithms at a glance.

First, Weka machine learning tool was used as the first approach then its results were recorded independently. Secondly, implementation of Python code in JupyterLab was conducted and its results recorded. The results obtained from both platforms were then compared. Before the classifier is built, there are data issues that need to be addressed in order to obtain data that is best for building the classifier. Data pre-processing, feature selection and class imbalance are problems that should be addressed before training a classifier.

Data collection

Secondary data was used for the experiment, a cyber security dataset entitled KDD99 Network Attack Dataset was downloaded from the UCI Machine Learning Repository. Network traffic logs were extracted, sampled and processed for training in Weka and JupyterLab. The original dataset contained 494,020 instances and 119 attributes.

KDD99 was preferred because of the following reasons: The capabilities of this dataset outweighs other datasets, recent datasets like ISCX 2012 and UNSWNB15 have no classification of the types of attack, thus it only provides binary classification unlike KDD99 which provides multi classification, a suitable option for this study. Secondly, the Defense Advanced Research Projects Agency (DARPA) 1998 and 1999 datasets are the most extensively used in experiments and are the most cited (Buczak, 2018). This is because they are public and its results can be compared to previous papers, ML models can only be compared is if the same data is used for training and testing (Niak, 2019)

The CSV format of the downloaded file format was then converted into ".arrf", a format that is understood by Weka. Primary data was later obtained through observations displayed in Weka, results obtained using Naïve Bayes were observed and then recorded for further analysis and the data obtained was compared with results from other experiments in other literature.

Data pre-processing

Missing Values and Inconsistent Data

Data exploration was done to identify and remove inconsistent data like records with missing values, or wrong type like character values in numeric fields.

Feature Selection

Attributes with the most predictive information was selected from the entire feature and visualized using heatmap.

3.3.2 Design of engine

Design of engine is a framework used to learn the structure of a scheme using previous data as the basis. It has various components, which includes the design of classification and its experimental flow.

Classification task design

Classification task was done by collecting relevant data for classifier learning process and testing process. The Waikato Environment for Knowledge Analysis (Weka), an open source tool for Machine Learning was used for this task. This is a tool that operates on most platforms; Linux, Windows and Mac. It is a graphical user interface which include; "The Explorer" "The Experimenter" and "The Knowledge Flow" which provides a good platform for data visualization. It has a variety of machine learning algorithms such as Bayesian Network, Support Vector Machine, C4.5 Decision Tree and K-Nearest Neighbor. It also supports a number of machine learning activities such as data pre processing clustering, classifier training and classifier testing.

The experiment flow

Figure 3.1 is a flowchart diagram showing systematic process for the experiment.

After the file containing the captured Network traffic has been converted into Attribute Relation File Format (ARFF) format it is then processed in Weka and goes through the following stages.

Experimental Steps

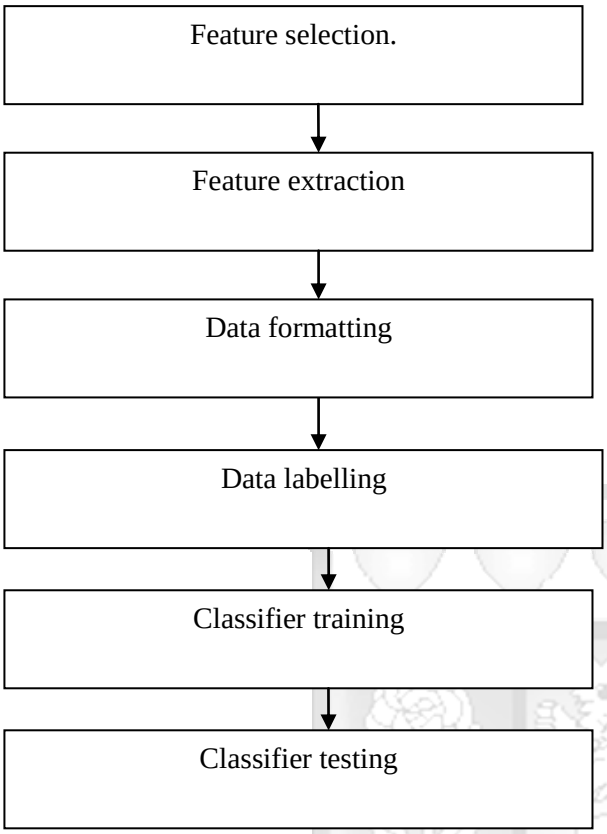


Figure 3.1: Experimental stages in building the classifier

A. Feature selection.

In this step, unique properties, which are only relevant for classification, are selected. This will help the classifier to perform faster and accurately. It is an effective data reduction approach, to extract the key attribute set from the original high-dimensional Network traffic data. Features with the most predictive information such as packet duration, packet length, inter arrival and packet protocol will be extracted from the feature set. Class imbalance happens when elements of a class are not classified even in their own class, which implies that an algorithm is not good enough.

B. Feature extraction

Feature extraction step follows after particular features have been selected. Extracted features are important components during classification process. Data that was in form of ".csv" format

was converted to .arff format before data analysis was done. To convert this data in "csv", ">" had to be added in order to redirect the output of the file to ".csv" file. This enables the file to be read by Weka or any common text file reader.

C. Data formatting

Once the file was converted to ".arrf" a format recognized by Weka, the data set was then loaded into Weka for processing. Weka was chosen since it is an open source tool that is readily available and has other built in classifiers.

Weka ARRF (Attribute-Relation File Format) is an ASCII file that describes instances sharing a set of attributes. An instance can be termed as rows while attributes can be termed as columns in a dataset.

The next step is labelling.

D. Data Labeling

This is a process of adding a class feature to extracted data.

The original dataset had 24 types of attacks and these types had to be classified into five classes namely: Normal, u2r, dos, r2l and probe. Labelling was done in excel before the dataset is processed in Weka tool since labelling is an important feature for classification. For IP traffic, the intention is to classify malicious traffic from normal traffic, and then they can be used as values of "normal" and "abnormal" where normal represents harmless and abnormal represents malicious traffic.

Where each instance is labelled as normal or abnormal, an appropriate learning algorithm is trained over the labelled data using Machine Learning classifier.

E. Classifier training

Once the data is pre processed, training process follows.

Weka GUI provides several approaches on how to train the classifier. Here we opted to use percentage split of 67% for training and 33% for testing. Weka splits the training data in 0.67 ratios of the total instances and uses the remaining part for training. The data set had 494,020

instances so it means 0.67* 494,020 = 330,993 will be used for training and 163,027 will be used to test the classifier. Allocating a higher proportion is important for training the classifier so as to enhance its accuracy. A well trained classifier gives better results for prediction. The process was repeated until all the training sets had been used to train and test the classifier and output of the process is the best average scheme.

F. Classifier testing

The trained classifiers were first tested independently and their results were recorded. This was done in order to compare how Naïve Bayes performs on different platforms. Performance parameters used to evaluate the performance of the classifiers are discussed in chapter 4.Weka outputs various statistics that evaluate the performance of the scheme.

3.4 Research quality

Research quality of this study adopted best practices from successful studies that have used Bayesian based algorithms for intrusion detection purposes. It was aimed at ensuring detection of anomalies and analysis in a seamless manner. Visualization graphs were used to give instant comparison view. The research was meant to improve on gaps identified in previous studies and inform future studies in an attempt to improve anomaly detection of Network intrusion in organizations.

3.5 Ethics considerations

The experiment was conducted with a sole purpose of testing the functionality of Naïve Bayes scheme. This experiment will only be performed using test machines mentioned in this research.

CHAPTER 4: SCHEME DESIGN AND ARCHITECTURE

4.1 Introduction

This chapter explains the steps that were to be used when designing the scheme. It discusses the functional and non functional requirements to be met by the scheme. System architecture, interaction diagrams were used to illustrate the interactions between users and the scheme.

4.2 Requirement analysis

This scheme was designed to meet some requirements that are listed in this chapter in order to be an effective intrusion detection system, particularly for anomaly detection. It encompasses both functional and non functional requirements to be met by the scheme.

4.2.1 Functional requirements

Functional requirements are aspects that affect the functions of the scheme directly (Dumas & Redish, 1999).A function can be termed as a specification of behaviour between outputs and inputs of a scheme. The functional requirements of the scheme are:

1. The scheme should be able to detect network anomalies.
2. The performance of this algorithm should be visualized using graphs for easy interpretation

4.2.2 Non functional requirements

Non functional requirements are functionalities that are not directly related to the core function of the scheme. They include:

1. The command line and GUI interfaces should be user friendly and there should not be any crashes in the system.
2. The scheme should be efficient in resource usage.
3. The scheme should be compatible with any PCs

4.3 Scheme architecture

The design of scheme consists of three main phases: pre processing, feature selection and classification. Data goes through an iterative cycle; each step of the entire process is revisited

repeatedly until the data is fit for the training process. Figure 4.1 shows the architectural design of the scheme.

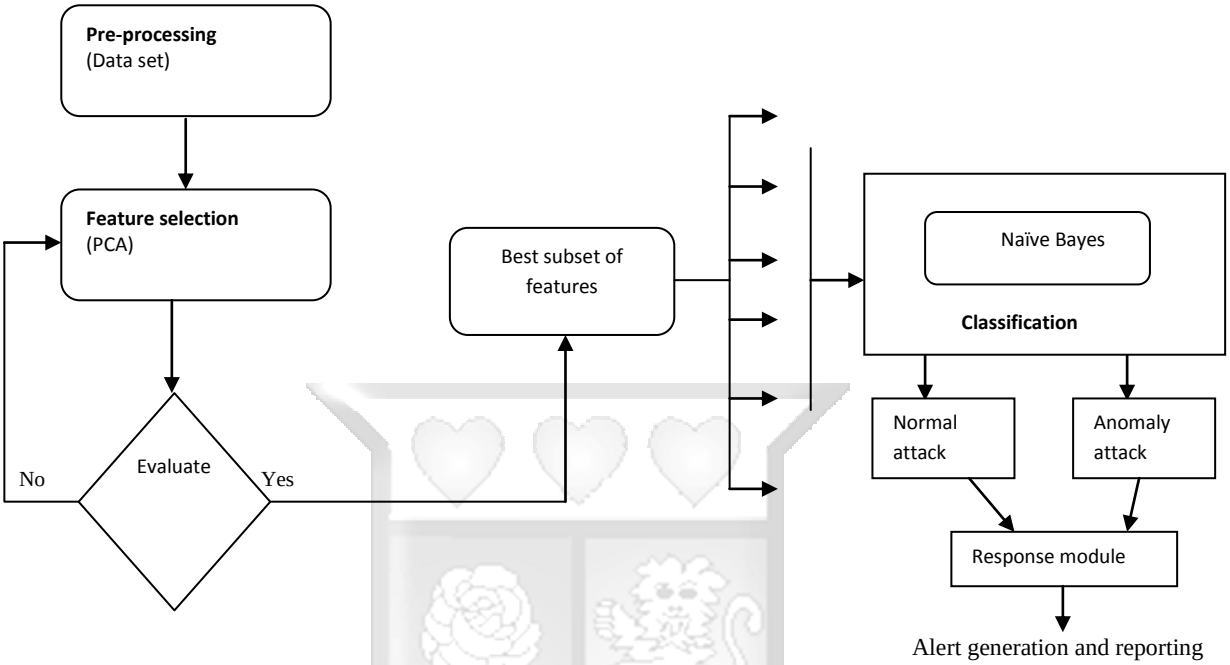


Figure 4.1: Design of scheme architecture (Own design)

4.3.1 Data collection

The data used was obtained from a dataset downloaded from a digital repository. The data collected was then divided for training and testing the data. A large portion was reserved for training while the smaller portion was reserved for testing.

4.3.2 Pre processing

Data pre-processing needed to be carried out in order to get rid of the redundant values and missing values. The entire data set had to be free from null values and redundant values before it is moved to feature selection step.

4.3.3 Feature selection process

The correct choice of features is a very important task during training and testing of the scheme. It determines how the scheme is going to perform. This is a process where some of the features were required to be dropped due to very high missing values and variables that had low correlation with intrusion activity. Variables that had low correlation with the target were dropped because it was not going to make any useful prediction. Features that had complete values and high correlation were picked to increase the accuracy, training ability and reduce data dimension before it is passed on to classification phase (Garthwaite & Jolliffe, 2002).

Construction of an engine classifier

The scheme was designed by using Naïve Bayes. Data cleaning was done to remove data imbalance, variances and to reduce computational complexity (Maybank, Wei & Weiming, 2008). The classifier was designed to classify the attack classes by use of rules, where classification of data was either classified as normal or anomaly. Data that matched with the rules were then picked for training. The scheme was tested with already labelled data to predict the accuracy.

4.4 Scheme Design Tools

This section comprises of a scheme design which is supported by a context diagram, use case diagram and a sequence diagram.

4.4.1 Context Diagram

Figure 4.2 indicates the relationship between the user and the scheme. The main inputs in the scheme are data pre-processing, classification, pattern matching while the output involves prediction output.

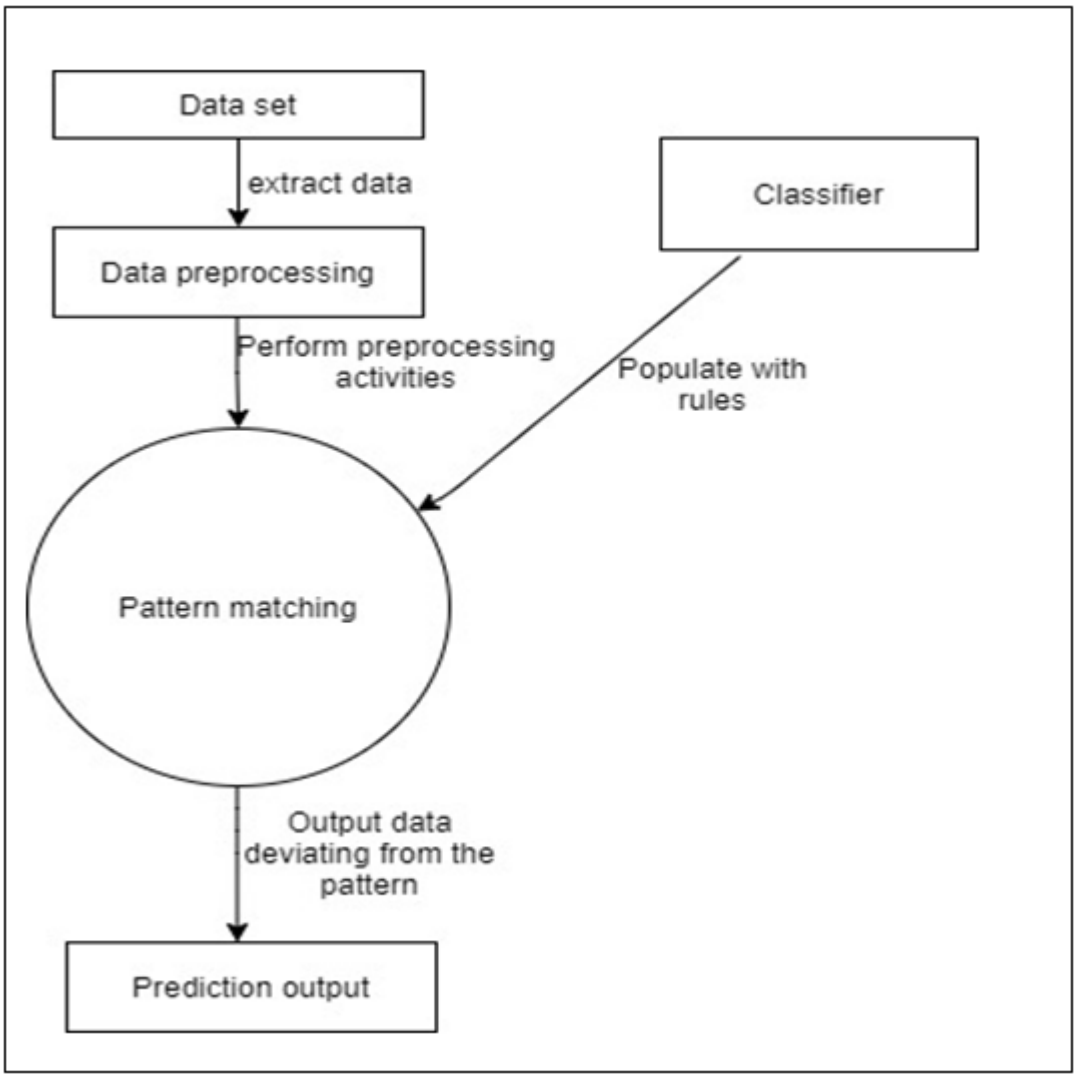


Figure 4.2: Context diagram

4.4.2 Use case Diagram

A use-case diagram shown in figure 4.3 shows the behaviour of the scheme. It depicts how two actors, system administrator and security administrator interact with the scheme to assist in anomaly prediction of the scheme. The classifier is trained to learn the behaviour if a normal attack by use of rules and when labelled data (which includes normal and anomaly data) is checked.

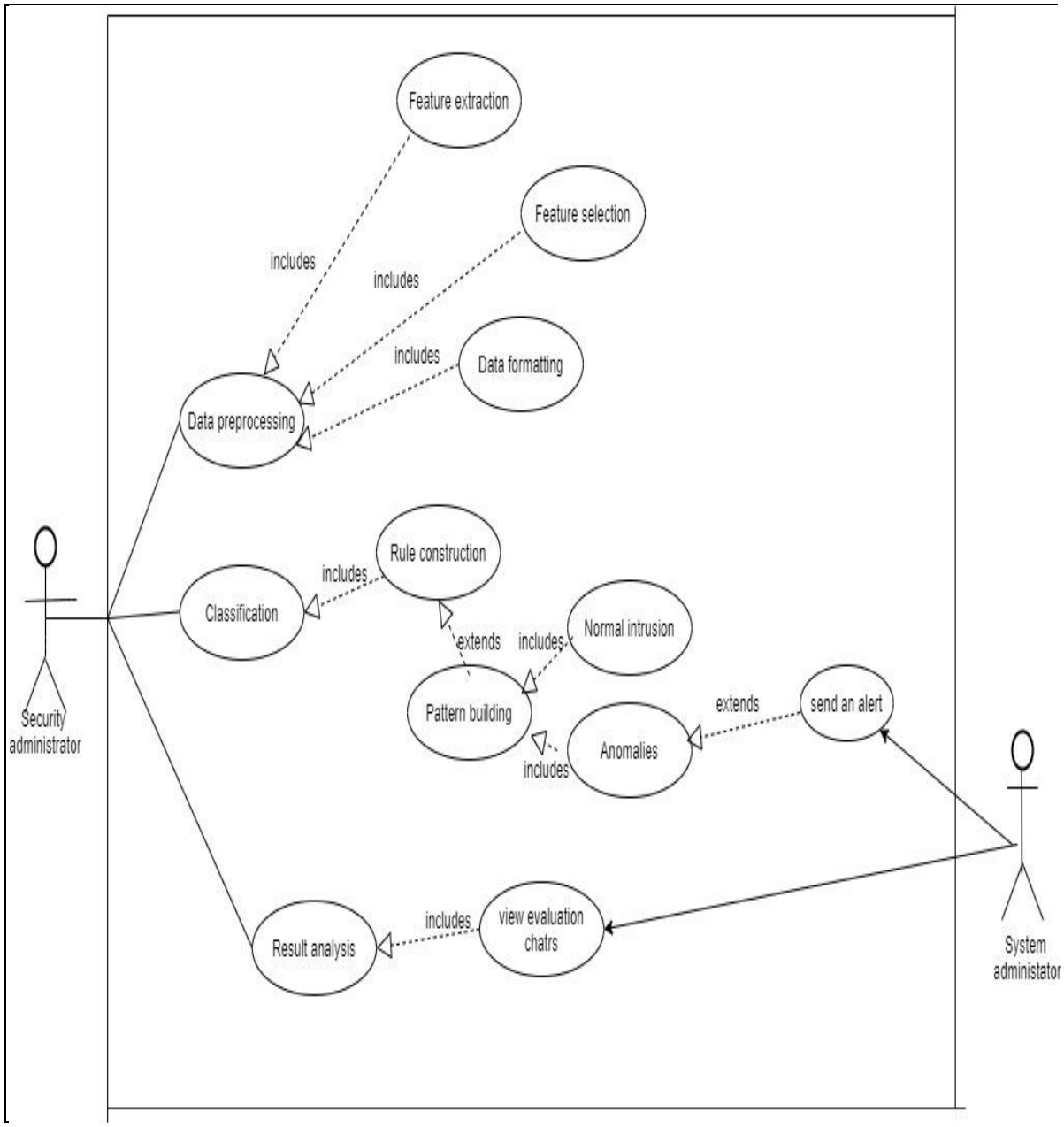


Figure 4.3: Use case diagram

4.4.3 Sequence Diagram

The sequence diagram in figure 4.4 illustrates flow of events for training and testing the scheme to predict anomaly intrusion on a network.

The first step is data pre-processing where data from the dataset is pre-processed, the most important features extracted are selected and formatted before eventually used for training the classifier. The classifier is populated with rules in order to build the scheme. The scheme is then tested using verified data to predict its accuracy.

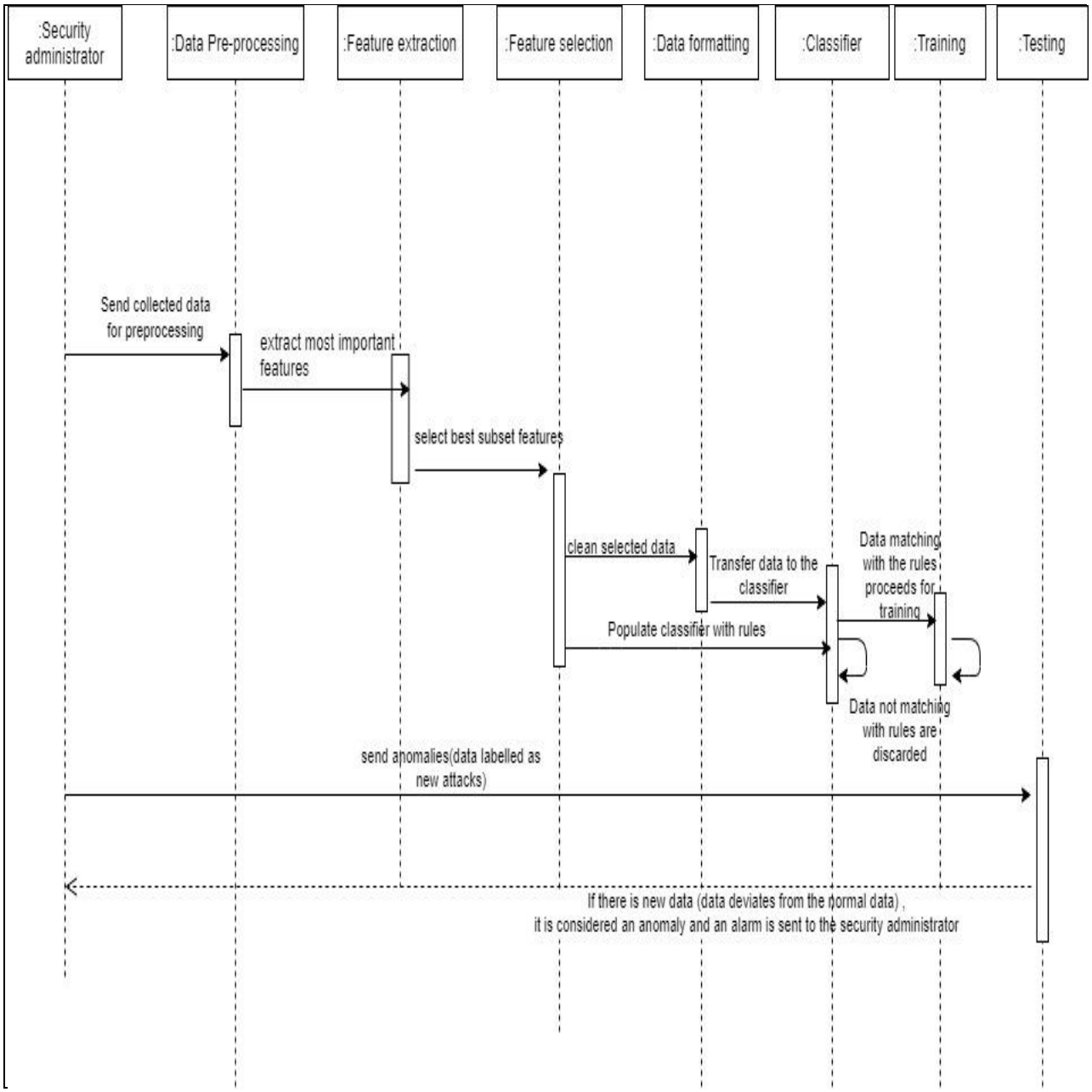


Figure 4.4: Systems sequence diagram

CHAPTER 5: IMPLEMENTATION AND TESTING

5.1 introduction

This chapter builds up on the design from the previous chapter. The tools, hardware and software components and the environment that was used to obtain the dataset are discussed as well as the experiment and process that was followed to build the scheme. The dataset had to be in memory during execution since it consisted of many attributes and instances. A detailed procedure on how the experiments were conducted and the steps that were followed is clearly explained.

5.2 Data set simulation description

KDD99 network attack dataset containing real intrusion attacks was obtained from a simulation that was performed by MIT Lincoln Labs The data set was downloaded from a UCI repository http://kdd.ics.uci.edu/databases/kddcup99/.A simulation environment was set up to acquire nine weeks of raw TCP dump data for a local-area network (LAN) simulating a typical U.S. Air Force LAN. They operated the LAN as if it were a true Air Force environment, but peppered it with multiple attacks. The raw data used for training, which was around four gigabytes, compressed binary TCP dump data was from the seven weeks of monitored network traffic. This was achieved by installing network traffic sniffer on a local air force base and then recording amounts and types of services such as SNMP,SMTP, telnet, FTP, HTTP, POP, domain, time amongst others. An isolated network was established and split into two and a router was placed in between where computers that were based inside were used to simulate an air force base while computers based outside were used to simulate the internet. The inside simulated a base of tens of machines and hundreds of users, while the outside simulated thousands of machines and the many users who make up the internet. Attack targets included sunOS systems, solaris systems, Linux and CISCO router with a sequence of TCP packets starting and ending at some well defined times, which data flows to and from a source IP address to a target IP address under a defined protocol. Connection was labelled as either normal, or an attack. Figure 5.1 shows the simulated network environment.

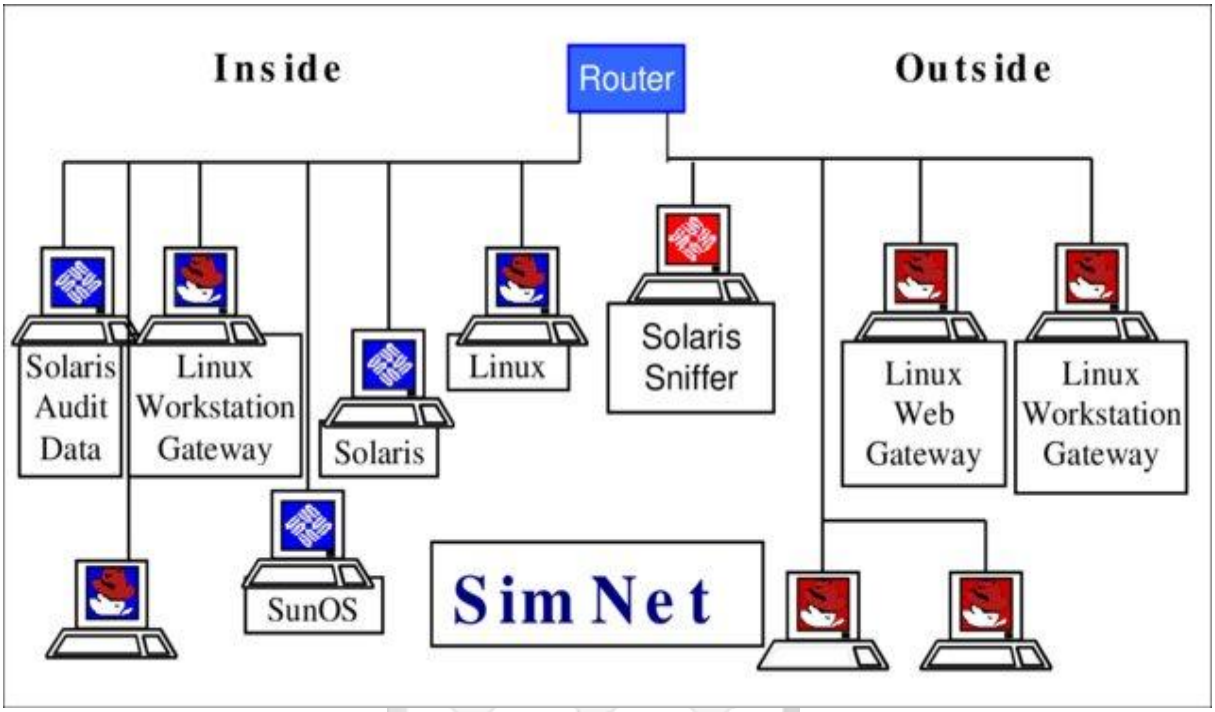


Figure 5.1: The simulated Network (Cunningham, Fried, Garfinkel, Graf, Kendall, Lippmann, Webster, Wyschogrod & Zissman, 1999).

5.3 Experiment setup.

The experiment was performed using a Core i5 processor 2.3 GHz, Windows 8 Pro 64-bit, and an 8 GB RAM. Two experiments using Weka machine learning tool and JupyterLab (Python) were performed to implement Naïve Bayesian classification.

5.3.1 Dataset Manipulation for experiment 1

Weka, a machine learning tool Version 3.8.1 was installed in the machine (Frank, Hall, Holmes, Kirkby & Pfahringer, 2009). Weka, was used for implementation of the scheme to train and test the classifier. It contains a set of machine learning algorithms that are applied directly on the data set. Naïve Bayes classifier was used for feature selection and classification.

Data preparation

The original dataset which was in .gz format was downloaded, extracted and saved into .csv file format.10% of the original dataset used contained data records that were not categorised, it consisted of 22 attack types and one normal type. The records were then labelled using the

following dictionary to form five class labels: Normal, U2R, DOS, Probe, and R2L. After labelling, the data was converted to .arff format that is recognized by Weka and saved in a folder.The labelled file saved in .arff format is then loaded into weka explorer to go through some manipulation stages such as data preprocessing and classification.

The procedure is captured in figure 5.2.

1. Start the Weka chooser

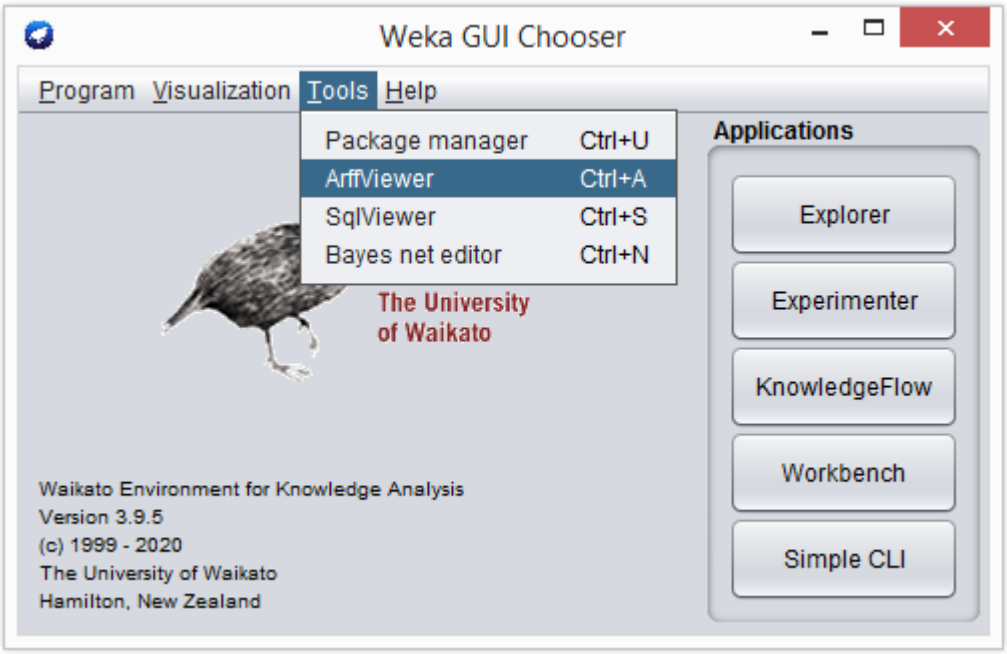


Figure 5.2: Weka GUI Chooser

2. Open the ARFF-Viewer by clicking "Tools" in the menu and select "ArffViewer", an empty ARFF-Viewer window will be displayed as shown in Figure 5.3

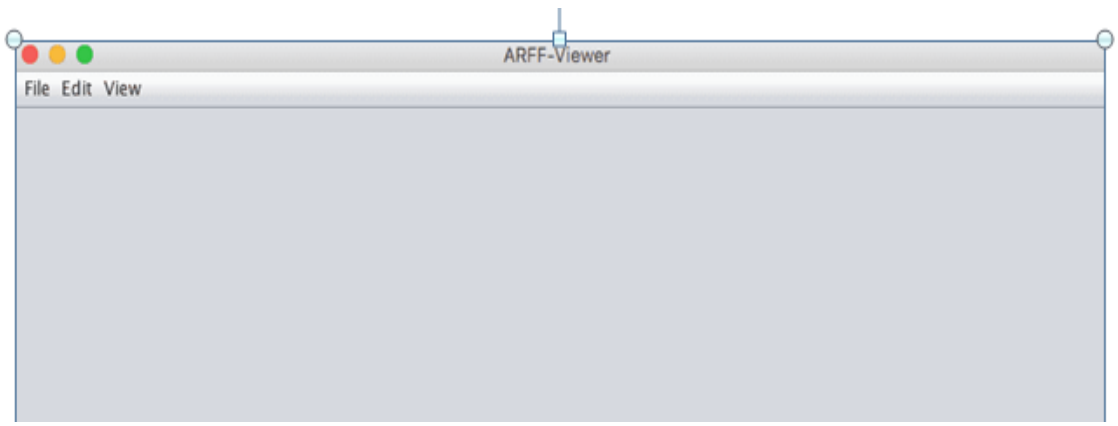
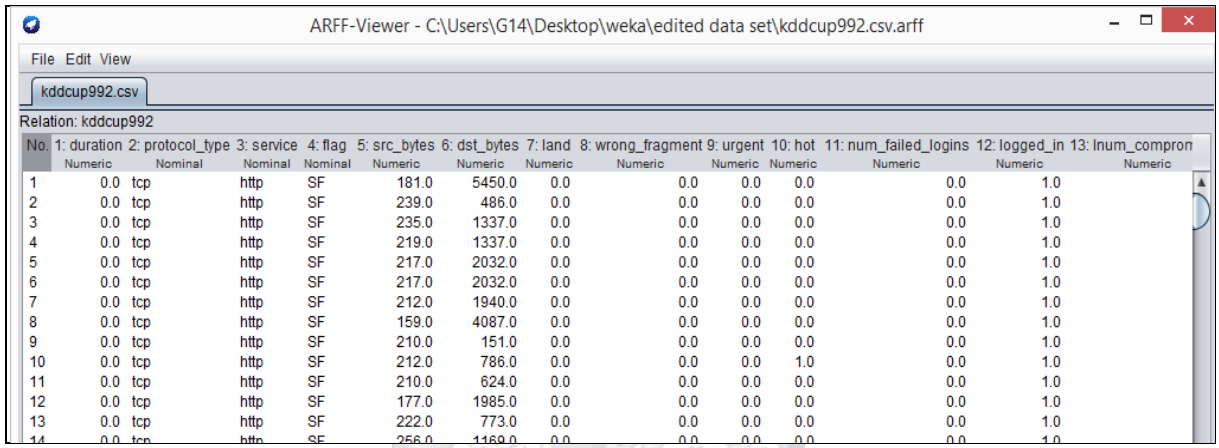


Figure 5.3 ARFF window viewer

3. Open the CSV file in the ARFF-Viewer by clicking the “File” menu and select “Open”. Navigate to your current working directory. Change the “Files of Type:” filter to “CSV data files (*.csv)”. Select your file and click the “Open” button.

The file opened is then automatically converted to .arff as shown in Figure 5.4



No.	1: duration	2: protocol_type	3: service	4: flag	5: src_bytes	6: dst_bytes	7: land	8: wrong_fragment	9: urgent	10: hot	11: num_failed_logins	12: logged_in	13: lnum_compron
	Numeric	Nominal	Nominal	Nominal	Numeric	Numeric	Numeric	Numeric	Numeric	Numeric	Numeric	Numeric	Numeric
1	0.0	tcp	http	SF	181.0	5450.0	0.0	0.0	0.0	0.0	0.0	1.0	
2	0.0	tcp	http	SF	239.0	486.0	0.0	0.0	0.0	0.0	0.0	1.0	
3	0.0	tcp	http	SF	235.0	1337.0	0.0	0.0	0.0	0.0	0.0	1.0	
4	0.0	tcp	http	SF	219.0	1337.0	0.0	0.0	0.0	0.0	0.0	1.0	
5	0.0	tcp	http	SF	217.0	2032.0	0.0	0.0	0.0	0.0	0.0	1.0	
6	0.0	tcp	http	SF	217.0	2032.0	0.0	0.0	0.0	0.0	0.0	1.0	
7	0.0	tcp	http	SF	212.0	1940.0	0.0	0.0	0.0	0.0	0.0	1.0	
8	0.0	tcp	http	SF	159.0	4087.0	0.0	0.0	0.0	0.0	0.0	1.0	
9	0.0	tcp	http	SF	210.0	151.0	0.0	0.0	0.0	0.0	0.0	1.0	
10	0.0	tcp	http	SF	212.0	786.0	0.0	0.0	0.0	1.0	0.0	1.0	
11	0.0	tcp	http	SF	210.0	624.0	0.0	0.0	0.0	0.0	0.0	1.0	
12	0.0	tcp	http	SF	177.0	1985.0	0.0	0.0	0.0	0.0	0.0	1.0	
13	0.0	tcp	http	SF	222.0	773.0	0.0	0.0	0.0	0.0	0.0	1.0	
14	0.0	tcp	http	SF	256.0	1169.0	0.0	0.0	0.0	0.0	0.0	1.0	

Figure 5.4: Conversion of .csv format to .arff format.

Data pre-processing

After saving the file in .arff format the file was then loaded directly into Weka Data pre-processing was done to remove issue of noisy data. Noisy data are data with large irrelevant and meaningless information that would cause problems in the subsequent processes. The data that had irrelevant information such as special symbols like “&”, “*” in wrong columns that should have only contained numerical and not characters, duplicated information such as attributes with same values were also removed. Rows containing missing data such as empty attributes were deleted and were not included during the classifier training. The reasons for dropping some features and selecting was due to the following reasons: variables with very high missing values, features mostly of the same value were dropped because the classifier would not learn anything from it. Also variables with very low correlation with the target were also ignored since it was not useful for the scheme. Pre- processing was performed as indicated in figure 5.5

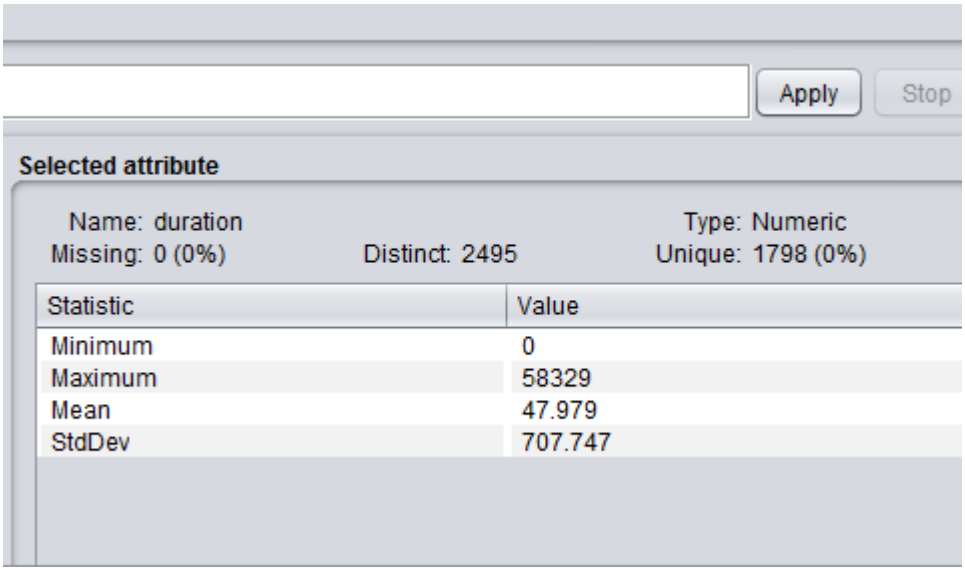


Figure 5.5: Data pre-processing.

Feature selection

The number of features was reduced by using filter method; Stratified Remove Folds was used to rearrange data to ensure each fold is a good representation of the whole so that each class comprises at least half of the instances.

Sampling using Stratified Remove Folds is shown in figure 5.6

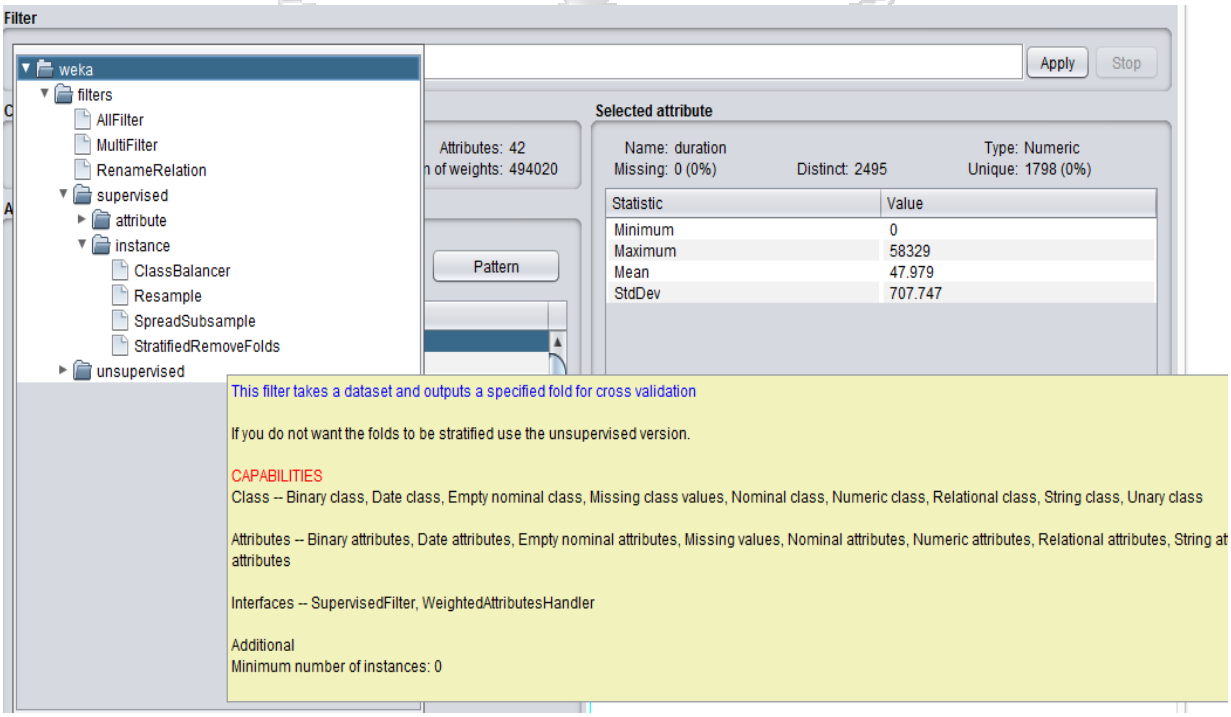


Figure 5.6: Feature selection using Stratified Remove Folds

Construction of classifier rules

Classification rules assist in defining which packets are to be selected for training. Rule action informs what the engine should do when it finds a packet that matches the rule criteria. Sample rules that were used to build the training scheme are as follows;

```
df.drop('num_root', axis = 1, inplace = True)

df.drop('srv_serror_rate', axis = 1, inplace = True)

pmap = {'icmp':0, 'tcp':1, 'udp':2}

df['protocol_type'] = df['protocol_type'].map(pmap)

fmap = {'SF':0, 'S0':1, 'REJ':2, 'RSTR':3, 'RSTO':4, 'SH':5, 'S1':6, 'S2':7, 'RSTOS0':8, 'S3':9, 'OTH':10}

df['flag'] = df['flag'].map(fmap)
```

Training and testing

After data had been pre processed and features selected, ‘k’ cross validation method was used. During this procedure a single parameter called k that refers to the number of groups that a given data sample is to be split into. The procedure is called k-fold cross-validation. When a specific value for k is chosen, it may be used in place of ‘k’ in the reference to the model, such as k=10 becoming 10-fold cross-validation. Data was split into training and testing set, 67% out of the total 494,020 instances was used for training the classifier and the remaining part of 33% was preserved for testing. Weka has various options for training a classifier but for this study percentage split of 67% was used. A large portion was allocated for training the classifier so that it improves the prediction accuracy of the scheme. More data used during training of the classifier increases the accuracy of the classifier. A classifier that is not well trained will yield to poor prediction results. The process of cross validation is as shown in figure 5.7

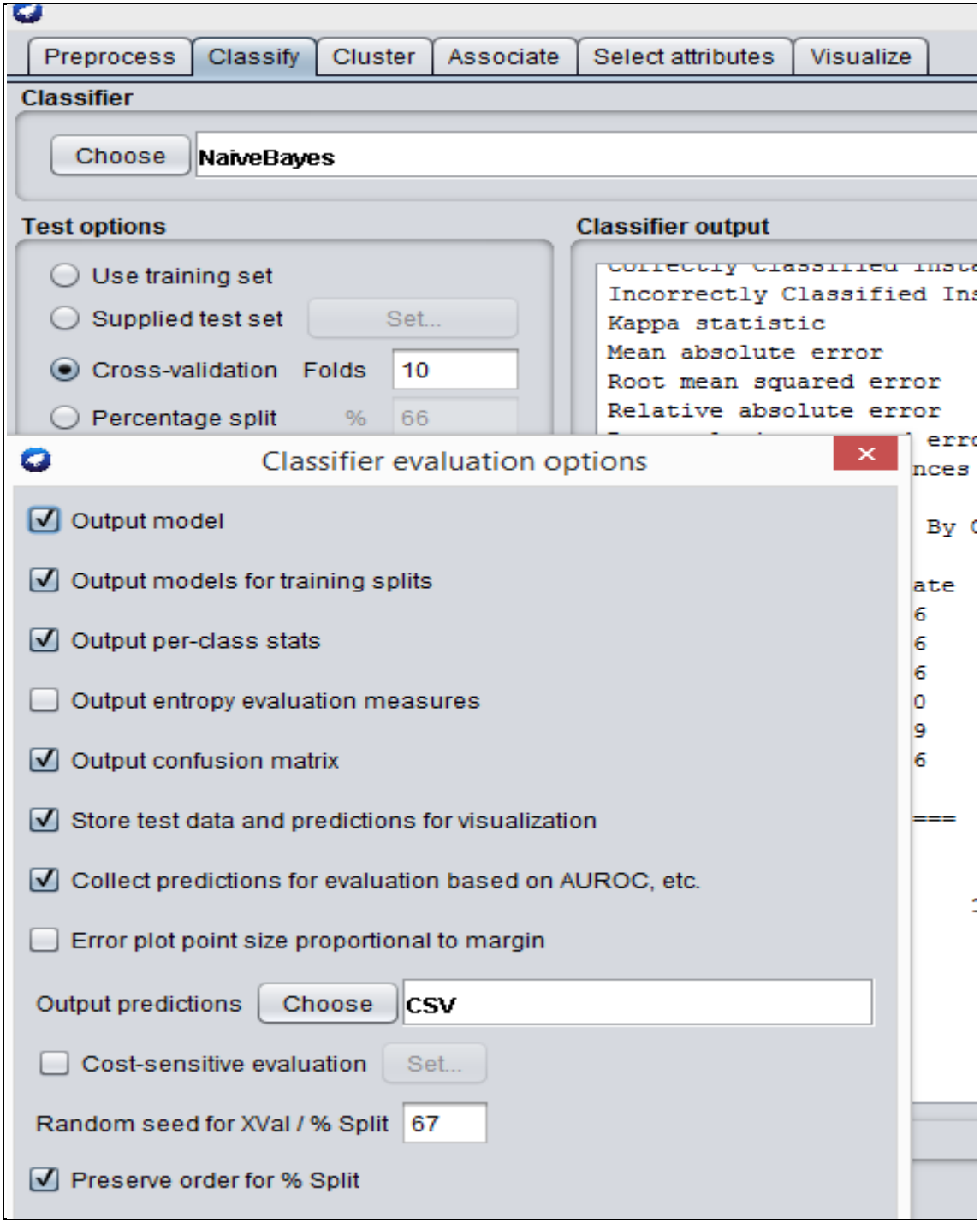


Figure 5.7: Cross validation process.

The time that was taken to build the model was 11 seconds as shown in figure 5.8

```
Time taken to build model: 11.01 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 17.87 seconds
```

Figure 5.8: Period used to build Naïve bayes

The time that was taken to build Support Vector Machine was 1101.03 seconds as shown in figure 5.9

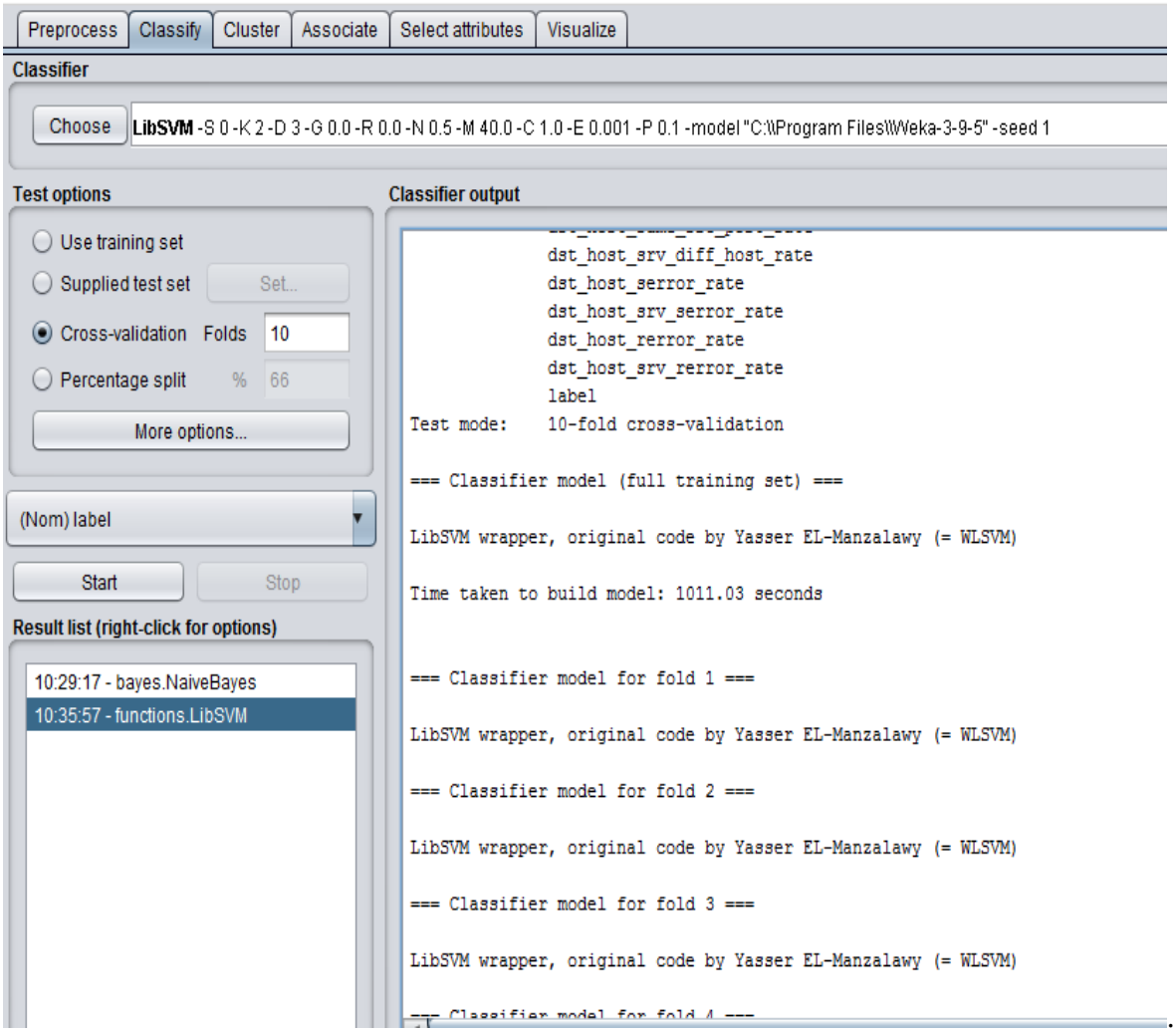


Figure 5.9 Period used to build Support Vector Machine

5.3.2 Dataset Manipulation for experiment 2

In the second experiment, Naïve Bayes classifier was implemented using Python 3.6, which included several python packages: ScikitLearn, Numpy, Pandas, Seaborn, Math, Itertools, Catplot and Matplotlib were used for plotting graphs. Software's that were used to run the program are Anaconda and Microsoft Excel.

Data pre-processing

Importing python libraries

First data libraries were imported such as numpy numerical library, data analysis library Pandas, and visualization library matplotlib.

Data was available in .CSV file format, read_CSV was used to load the data which returned the dataframe object and the available data was shown in figure 5.10

```
# Code: Importing libraries and reading features list from 'kddcup.names' file.

import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import time

# reading features list
with open("kddcup.names.txt", 'r') as f:
    print(f.read())
```

Figure 5.10: Importing python libraries

The data set was extracted as shown in figure 5.11

```
path = "kddcup.data_10_percent.gz"
df = pd.read_csv(path, names = columns)

# Adding Attack Type column
df['Attack Type'] = df.target.apply(lambda r:attacks_types[r[:-1]])
df.head()
```

Figure 5.11: Reading from .CSV file

Data labelling

After extracting the dataset from the file (‘kddcup.data_10_percent.gz’) class matching was done by labelling the attack types. There were 22 attack types in the original data set, the attack types which belonged to DoS, Probe, R2L and U2R attack classes were clustered into four categories i.e. dos, normal, probe, r2l and u2r. A column for attack type was added to the dataset.

The attack column is mapped and changed to attack class

1. normal: normal
2. dos : back ,land, Neptune, pod, smurf, teardrop
3. u2r: buffer_overflow, loadmodule, perl, rootkit
4. r2l:ftp_write, guess_passwd, imap, multihop, phf, spy, warezmaster, warezclient
5. probe: ipsweep, nmap, portsweep, satan

Matching of the attacks with the class types is shown in figure 5.12

```
attacks_types = {
  'normal': 'normal',
'back': 'dos',
'buffer_overflow': 'u2r',
'ftp_write': 'r2l',
'guess_passwd': 'r2l',
'imap': 'r2l',
'ipsweep': 'probe',
'land': 'dos',
'loadmodule': 'u2r',
'multihop': 'r2l',
'neptune': 'dos',
'nmap': 'probe',
'perl': 'u2r',
'phf': 'r2l',
'pod': 'dos',
'portsweep': 'probe',
'rootkit': 'u2r',
'satan': 'probe',
'smurf': 'dos',
'spy': 'r2l',
'teardrop': 'dos',
'warezclient': 'r2l',
'warezmaster': 'r2l',
}
```

Figure 5.12: Dictionary for the attack types

Feature selection

Selection of features with highly correlated variables was done and the variables were ignored for analysis. Such columns were dropped and columns with unique values were retained. Features mostly of the same value were dropped because the classifier would not learn anything from it. Feature selection is illustrated in figure 5.13.

```
#identifying categorical features
def det_attribute_type(dataset):
    numeric_cols = dataset._get_numeric_data().columns
    categorical_cols = list(set(dataset.columns) - set(numeric_cols))
    categorical_cols.remove('attack')
    categorical_cols.remove('target')
    return categorical_cols
cat_columns = det_attribute_type(df)
cat_columns
```

Figure 5.13: Retained columns

After features have been selected, data is cleaned where variables with missing data is cleaned as illustrated in figure 5.14

Cleaning the data

```
#dropping columns with missing values
df = df.dropna('columns')
df = df[[col for col in df if df[col].nunique() > 1]]
```

Figure 5.14: Dropped columns

Sampling

After pre processing and feature selection training is done using stratifiedK fold method. StratifiedKFold shuffles the data before splitting, after that it splits the data into ‘k’ splits parts. Data was split into two, training and testing set. 67% out of the total 494,020 instances was used for training the classifier and the remaining part of 33% was preserved for testing. Figure 5.15 shows the method that was used for cross validation.

```
acc_score = []
folds = StratifiedKFold(n_splits=10)
for train_index, test_index in folds.split(X,y):
    p2 = []
    X_train, X_val = X.iloc[list(train_index)], X.iloc[list(test_index)]
    Y_train, Y_val = y.iloc[list(train_index)], y.iloc[list(test_index)]
    classifier = GaussianNB()
    classifier.fit(X_train, Y_train)
    predictions = classifier.predict(X_val)
    print("err: ", accuracy_score(Y_val, predictions))
    print("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~")
    acc_score.append(accuracy_score(Y_val, predictions))
np.mean(acc_score)
```

Figure 5.15 Cross validation using stratifiedK fold method

Training and testing of the data set of the scheme is shown in figure 5.16

```
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.33, random_state=42)
```

```
#Our Gaussian Naive-Bayes model can then be saved for deployment into the selected platform
selected_classifier = GaussianNB()
selected_classifier.fit(X_train, y_train)
predictions = selected_classifier.predict(X_test)
```

Figure 5.16: Training and testing of the scheme

5.3.3 Performance evaluation for experiment 1

To evaluate the results of classifier we use evaluation criteria such as confusion matrix, true positive rate, false positive rate, and classifier's accuracy.

Confusion Matrix*:* This may be used to summarize the predictive performance of a classifier on test data. It is commonly encountered in a two-class format, but can be generated for any number of classes. A single prediction by a classifier can have four outcomes, which are displayed in table 5.17.

Table 5.17: Confusion matrix

		Predicted Class	
		Positive	Negative
Actual Class	Positive	TP	FN
	Negative	FP	TN

True positive (TP): Number of positive instances correctly predicted.

False negative (FN): Number of positive instances wrongly predicted as negative.

False positive (FP): Number of negative instances wrongly predicted as positive

True negative (TN): Number of negative instances correctly predicted.

When the counts in the confusion matrix are expressed as percentages, we get other related measures:

True positive rate (TPR) or sensitivity: fraction of positive instances predicted correctly.

True negative rate (TNR) or specificity: fraction of negative instances predicted correctly.

False positive rate (FPR): fraction of negative instances wrongly predicted as positive.

False negative rate (FNR): Fraction of positive instances correctly predicted

Precision: fraction of records that actually turn out to be positive in the group the classifier has declared as positive. The higher the precision is, the lower the number of false positive errors committed by the classifier.

Recall: Fraction of positive instances correctly predicted by the classifier. Its value is equivalent to true positive rate. The higher the value of recall the fewer the number of instances misclassified as negative.

F-Measure: Represents the mean between recall and precision

Performance parameters

Performance parameters of the experiment conducted using Weka is summarized in figure 5.18

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

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.860    0.007    0.966      0.860    0.910      0.892    0.995     0.965     normal
                 0.944    0.014    0.007      0.944    0.015      0.083    0.994     0.084     u2r
                 0.939    0.010    0.997      0.939    0.967      0.866    0.985     0.996     dos
                 0.353    0.001    0.469      0.353    0.403      0.406    0.982     0.338     r2l
                 0.880    0.056    0.113      0.880    0.201      0.305    0.989     0.583     probe
Weighted Avg.    0.922    0.010    0.983      0.922    0.949      0.865    0.987     0.985
```

Figure 5.18: Performance parameters displayed by Weka

Confusion matrix obtained from Weka experiment is shown in figure 5.19

```
=== Confusion Matrix ===

     a      b      c      d      e   <-- classified as
 27523   1938    317    141   2067 |      a = normal
     0     17      0      1      0 |      b = u2r
   891     37 121475      4   6938 |      c = dos
    15    196      0    129     25 |      d = r2l
    62     88      7      0   1156 |      e = probe
```

Figure 5.19: confusion matrix from Weka experiment.

The Receiver Operating Characteristic (ROC) Curve

A receiver operating characteristic (ROC) curve is a graphical approach for displaying the trade off between true positive rate and false positive rate of a classifier. The y-axis represents the true positive rate while the x-axis represents the false positive rate. Each point along the curve corresponds to one of the schemes induced by the classifier.

A classifier located closer to the upper left corner of the diagram performs better while one that makes random guesses reside along the diagonal connecting the points (0,0) and (1,1). Random guessing means that a case is classified as a positive class with a fixed probability irrespective of its attribute set.

Generating a ROC curve

From the results obtained, the values were used to rank the predictions. These outputs correspond to probabilities generated by Naïve Bayes classifier. Y axis represents true positive rate while X axis represents false positive rate.

The performance of the experiment were plotted in 5 ROC curves since the data is a multi class.ROC curves were plotted for Normal,dos,probe,r2l and u2r.

Figure 5.20 shows the ROC curve for the ‘normal’ class

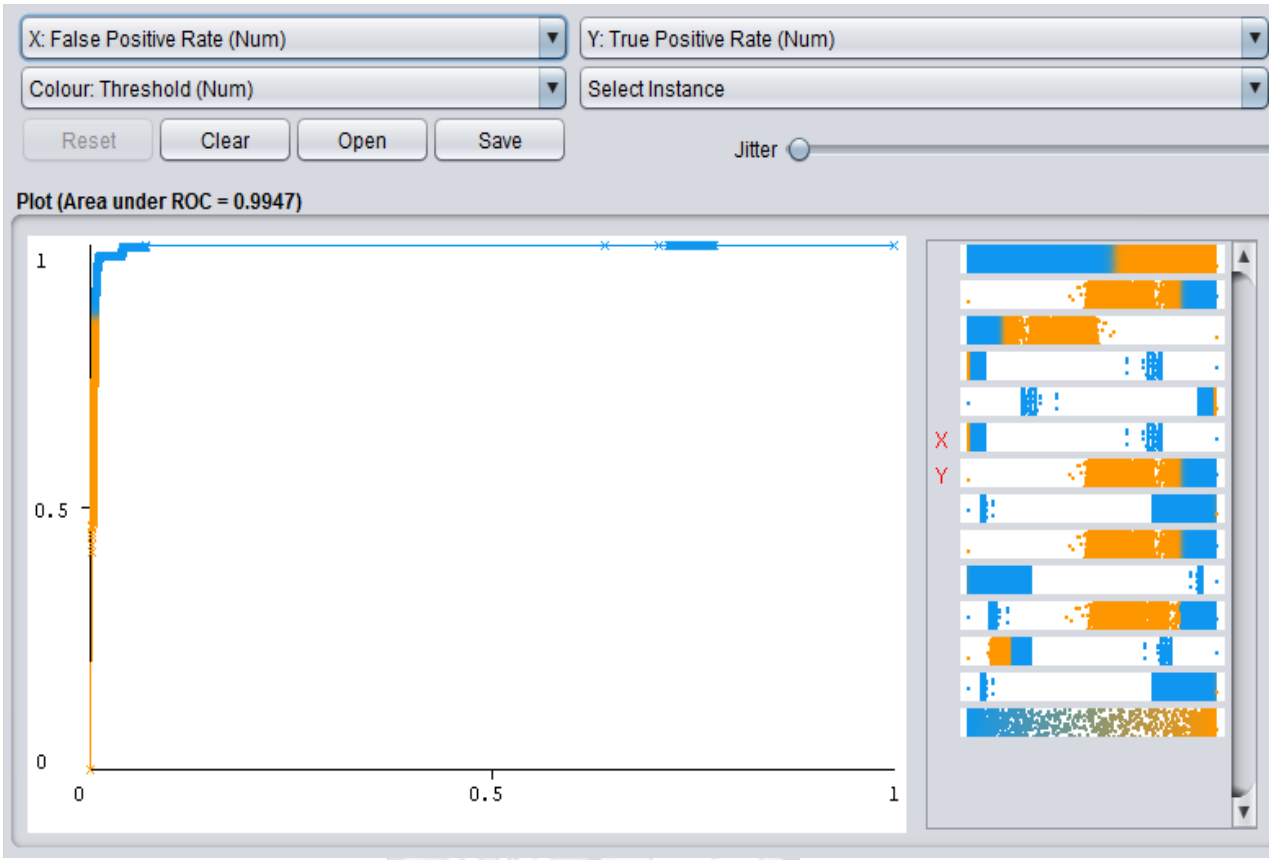


Figure 5.20: ROC curve for the ‘normal’ class type

Figure 5.21 shows the ROC curve for the ‘u2r’ class

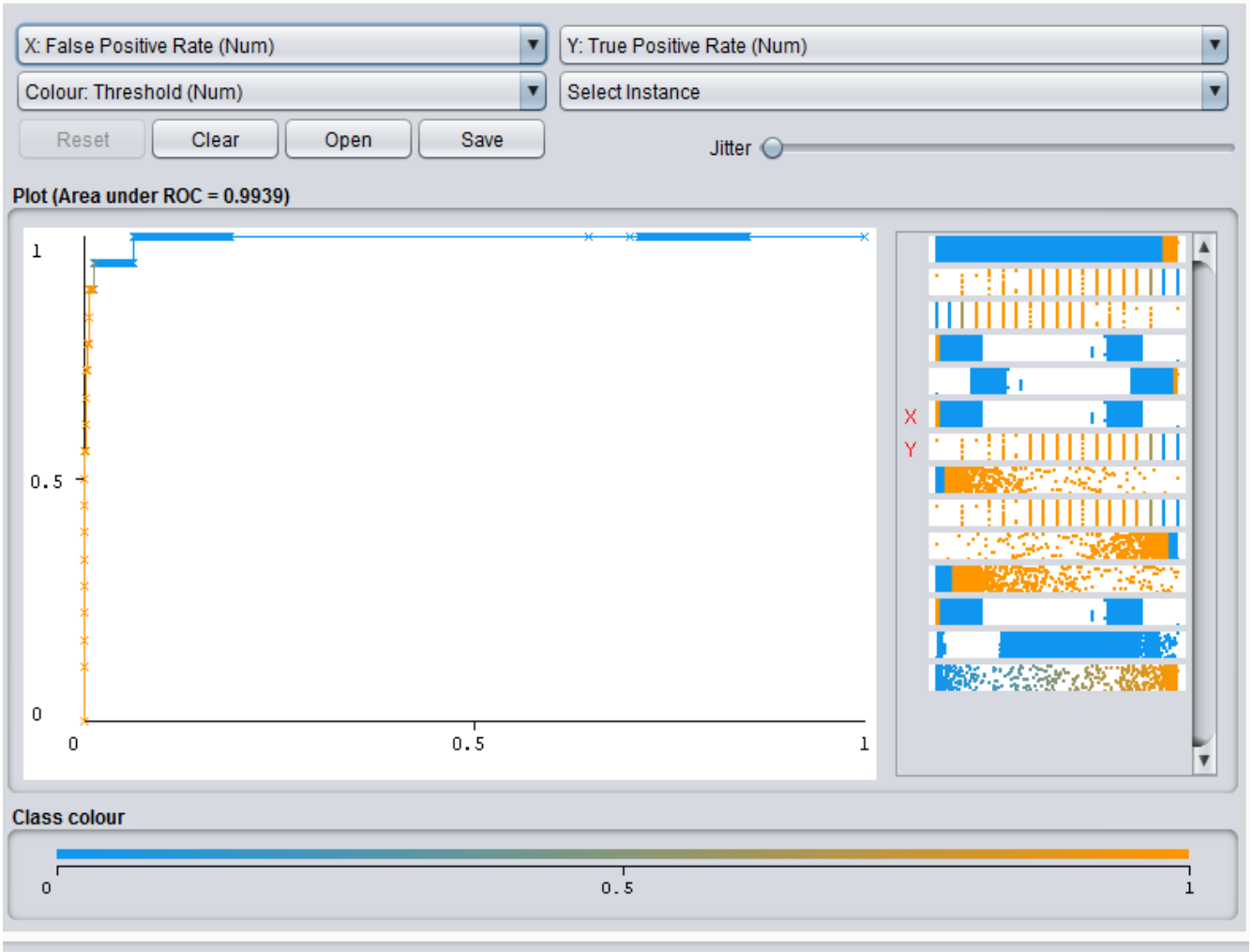


Figure 5.21: ROC curve for the ‘u2r’ class type

Figure 5.22 shows the ROC curve for the ‘dos’ class

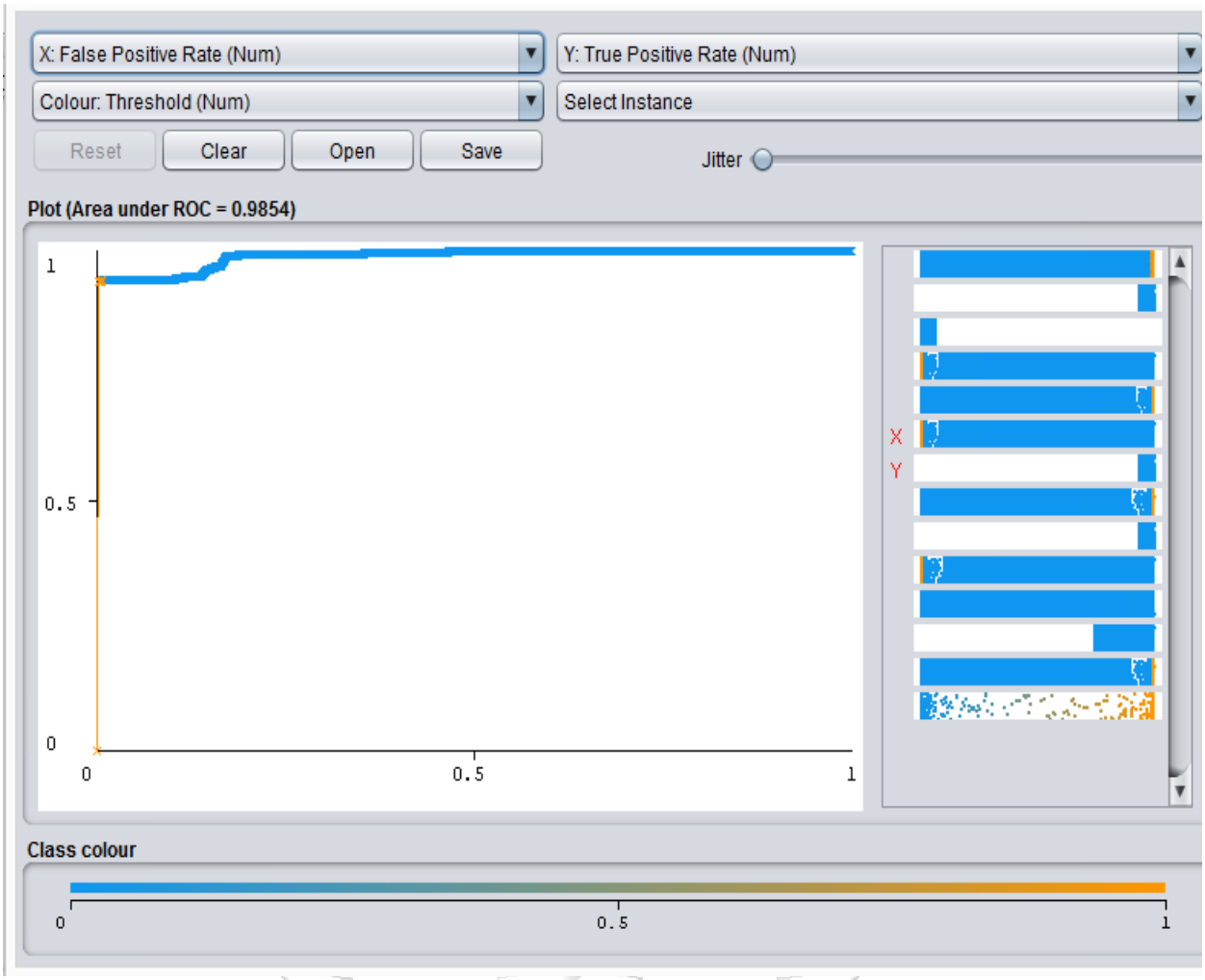


Figure 5.22: ROC curve for the ‘dos’ class type

Figure 5.23 shows the ROC curve for the ‘r2l’ class

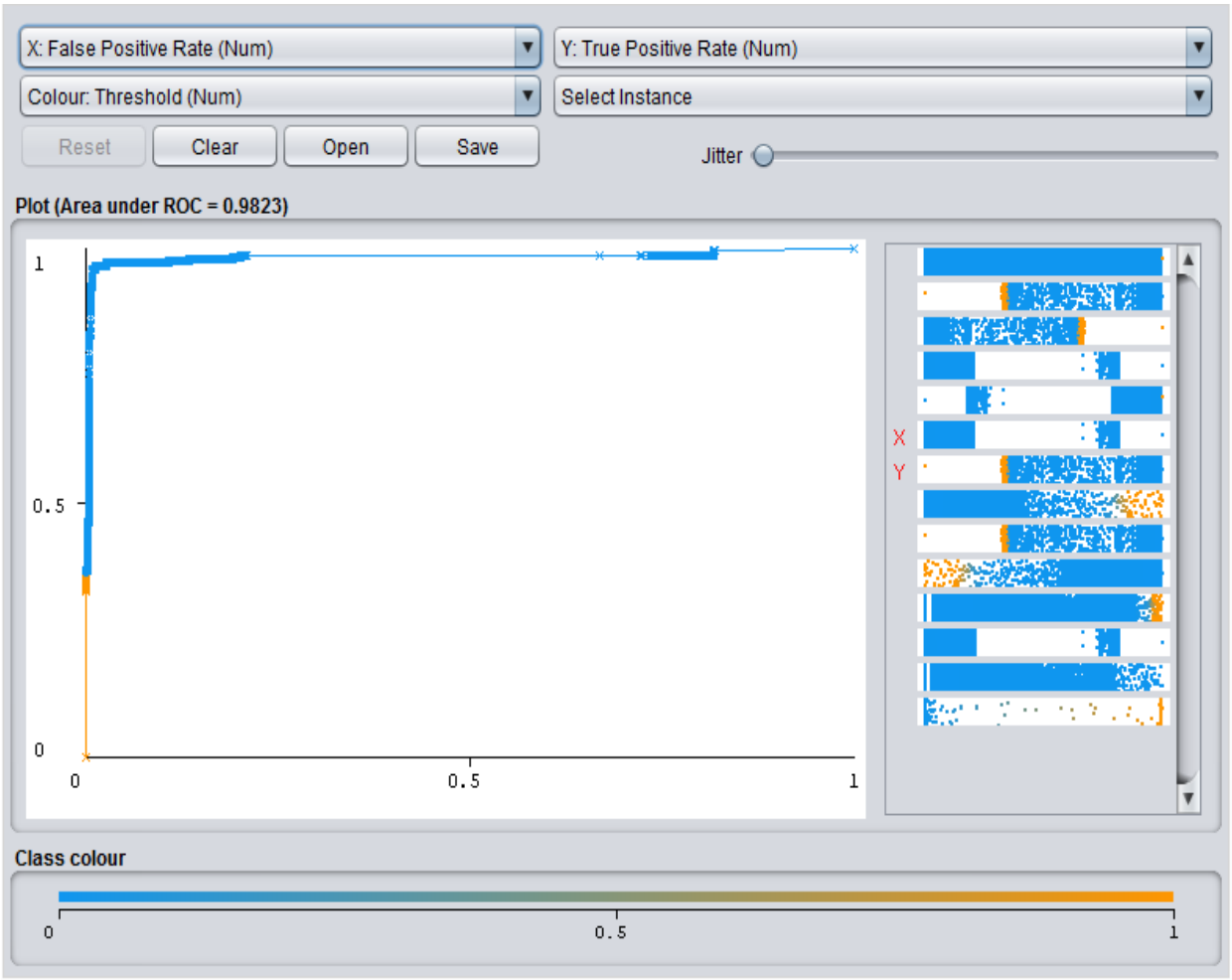


Figure 5.23: ROC curve for the ‘r2l’ class type

Figure 5.24 shows the ROC curve for the ‘probe’ class

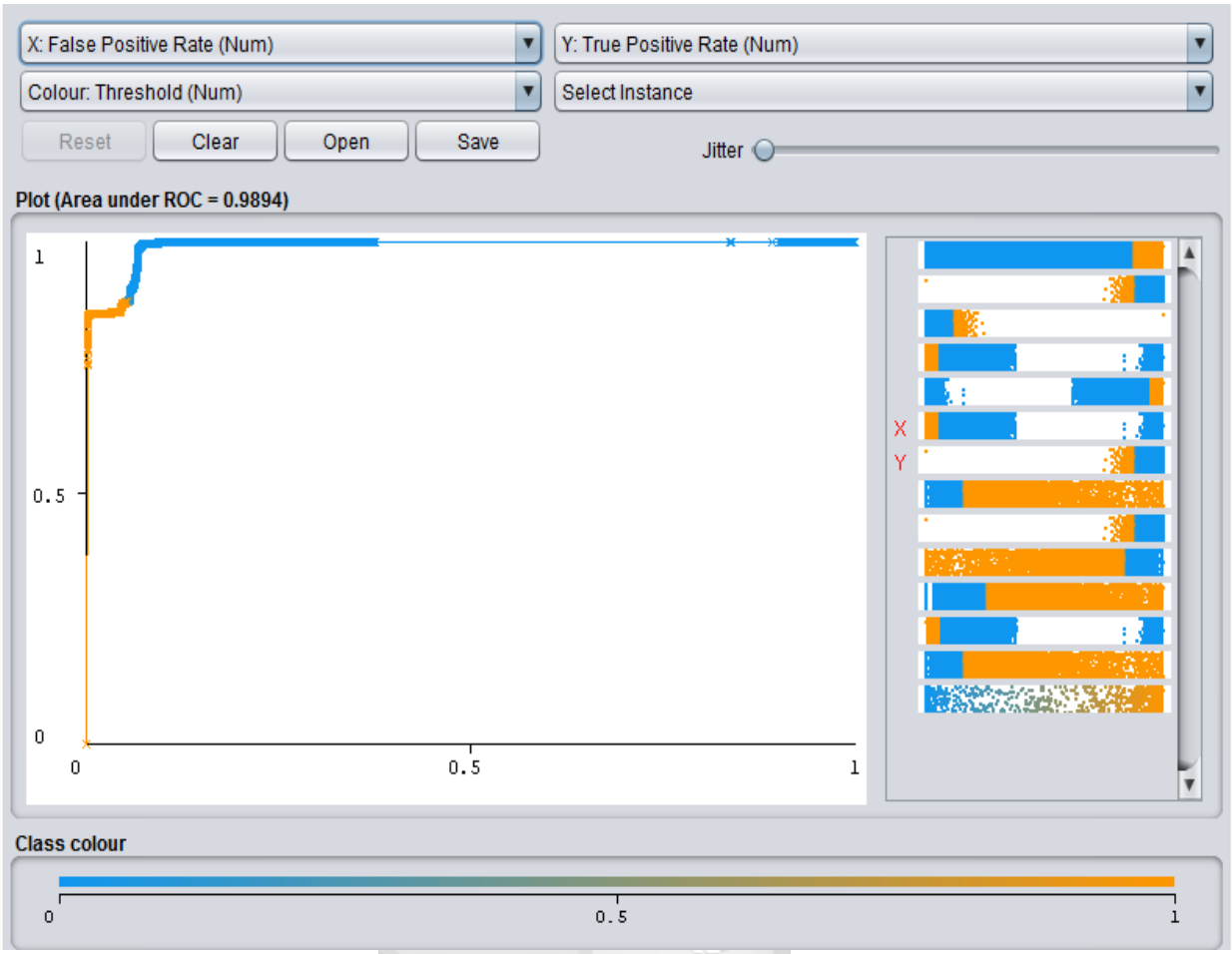


Figure 5.24: ROC curve for the ‘probe’ class type

Area under ROC curve

Area under ROC curve (AUC) is another approach used to compare performance of different schemes on average. An area under curve value of 1 means the scheme is perfect. If the scheme simply performs random guessing, then its AUC would be equal to 0.5. The larger the AUC value the better the scheme. AUC performances of the class types are as shown below.

1. Normal- 0.9947
2. u2r-0.9939
3. dos-0.9854
4. r2l-0.9823
5. probe-0.9894

Visualization

End results were displayed and the attributes seen with the most anomalies or normal attacks as indicated in different colours. Figure 5.25 shows the distribution of the attacks.

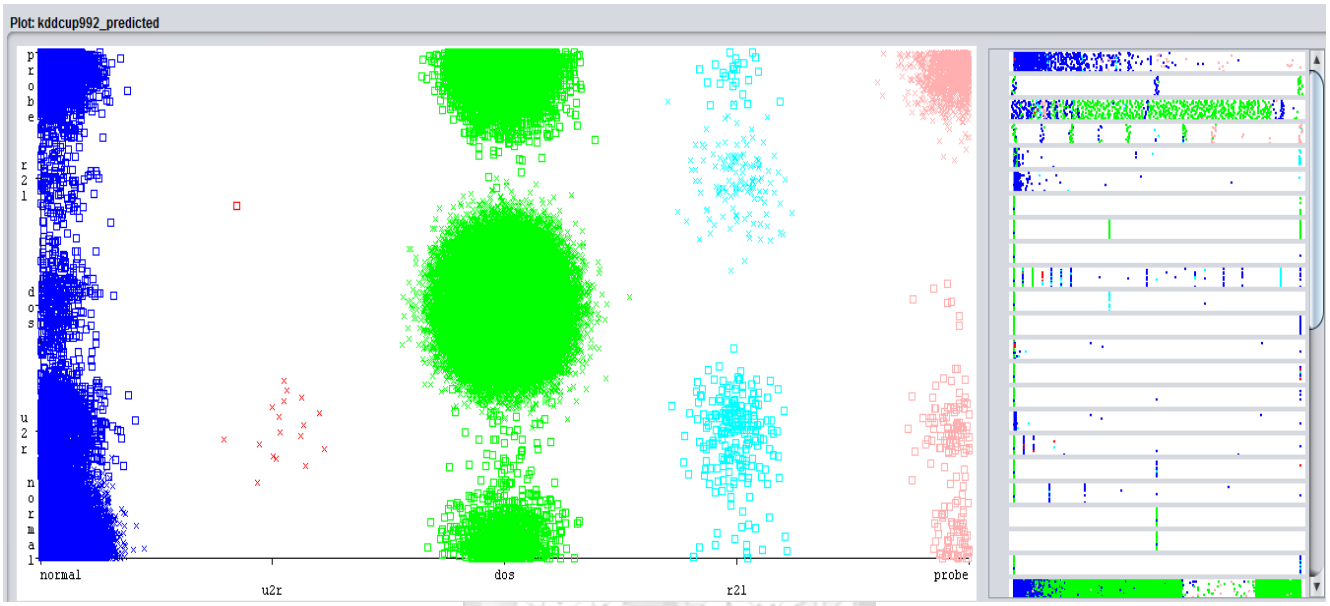


Figure 5.25: A representation of normal and anomaly attacks.

5.3.4 Performance evaluation for experiment 2

Performance parameters of the experiment conducted using Python is summarized in figure 5.26

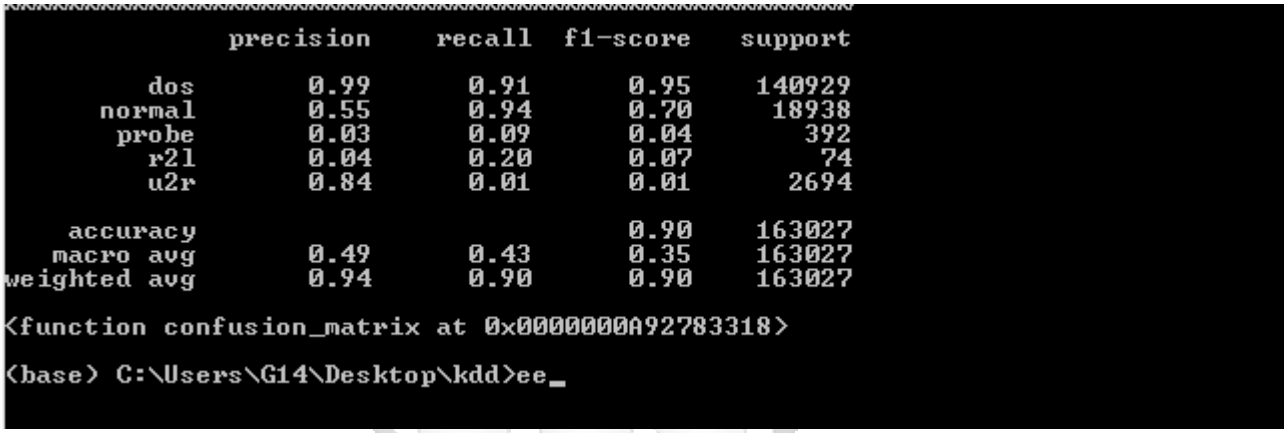


Figure 5.26: Performance parameters displayed by Anaconda prompt.

Confusion matrix obtained from JupyterLab (Python) experiment is shown in figure 5.27

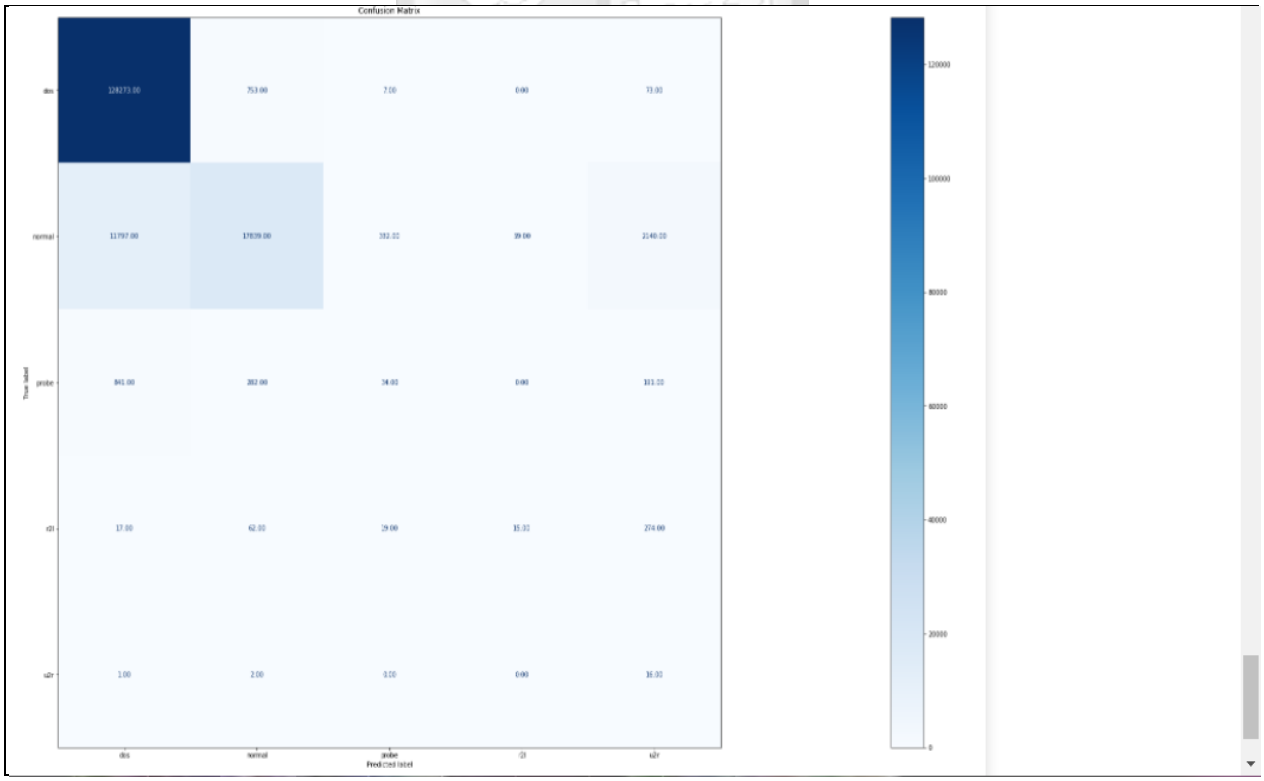


Figure 5.27: confusion matrix from JupyterLab (Python) experiment

The protocol that was predicted to experience many attacks was ICMP followed by TCP and lastly UDP. Protocol type distribution is shown in figure 5.28

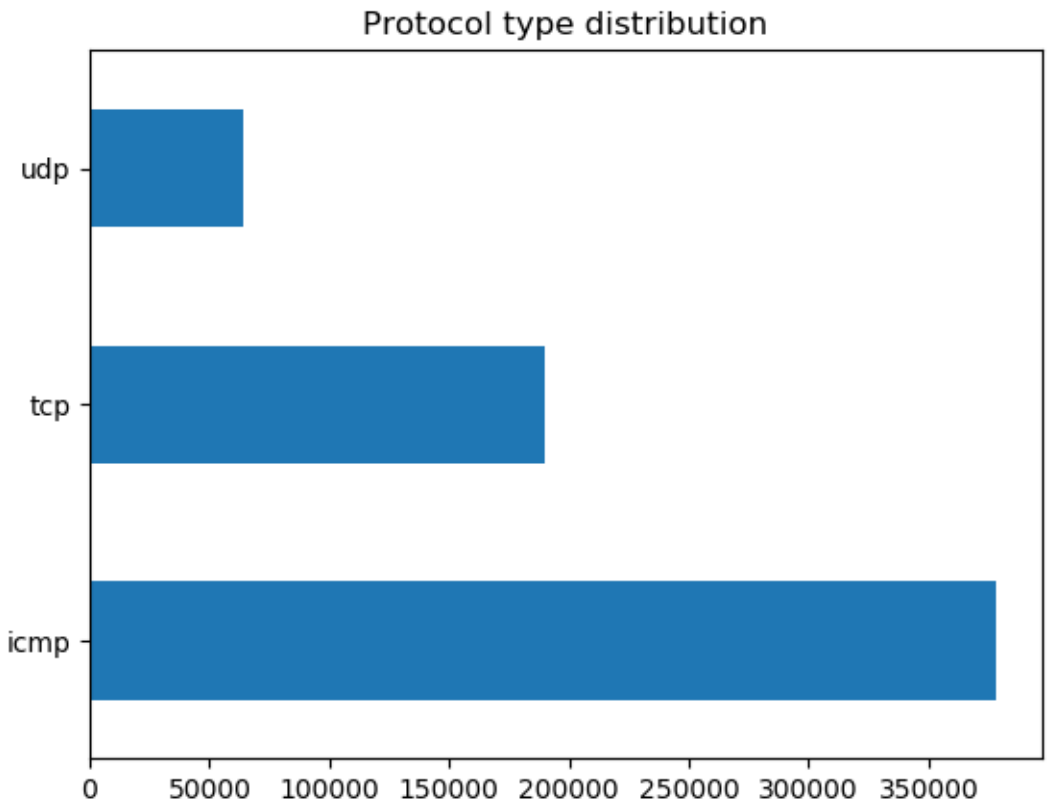


Figure 5.28: A representation of protocol type distribution.

5.3.5 Summary of results

From the results obtained by Weka and Python, the variance of 2% in accuracy is attributed to the methods of cross validation used. In Weka, a simple Kfold cross validation was used, the dataset was randomly shuffled and then split into parts in order to evaluate the performance. In Python stratifiedK fold cross validation was used which is a variation of KFold. First, Stratified process does the rearranging of the data to ensure each fold is a good representative of the whole and then it splits the data into training and testing parts.

From the two experiments, best performance is noted especially for DoS attacks as this is depicted from a true positive of 93%, precision rate of 99% and F1 score of 96%.However it performs poorly for r2l attacks as shown from its true positive of 35% precision rate of 46% and F1 score of 40%. This classifier is best especially for prediction of DoS attacks.

CHAPTER 6: DISCUSSION OF KEY FINDINGS

6.1 Introduction

The purpose of this study was to understand the common threats, vulnerabilities and attacks on the network and then look at the various approaches used to detect intrusions and then design a prototype that can provide a solution by predicting intrusions of bad connections. This was done to make it possible to detect intrusions that are unseen. This scheme is an ideal approach because it makes predictions as the classifier is able to learn from observations when exposed to more observations unlike signature based which searches for attacks using known patterns.

6.2 Objective one

The first objective was to identify common network insecurities that face the network such as threats, vulnerabilities and attacks. Common threats were identified as intentional or unintentional such as human accidental errors and omissions, hackers who are a big threat and insider threats like previous employees who may have left the organization dissatisfied. Vulnerabilities include viruses, passwords that may have been exposed, improper configurations and poor programming practices that render the network vulnerable. Unsecure applications which results from poor programming practices have become a major contribution to increased attacks. This is because hackers focus on weak points where they need not to spend much time attacking it.

6.3 Objective two

The second objective was to review existing techniques and approaches used in the field of network intrusion detection. There are many techniques used for network intrusions, this is due to emerging attacks caused by high increase in internet use. Various approaches have been reviewed and discussions of how they have been implemented in previous research areas have been made. From the analysis, Bayesian based networks compared to other approaches such as decision trees, neural networks and Support Vector Machine is an ideal approach in solving intrusion problems, especially for anomaly attacks. This is because of its dynamic mechanism, a feature that is important since there are always new attacks that are constantly changing day by day which is applicable for anomaly detection. Bayesian based networks are also known for its strong independence, probability of an attribute doesn't affect the probability of another

hence they are considered as powerful especially when reasoning under uncertainty. They are best in predicting outcomes with high interrelated variables.

6.4 Objective three

The third objective was to design, develop and test a prototype based Bayesian networks for detection of intrusions. Upon development, the scheme was implemented using two approaches: Weka machine learning tool and Jupyterlab (Python) was used for implementation. The classifier performed well in both platforms. When implemented using Weka, an accuracy of 92% was achieved while implementation using Python achieved 90%. This scheme was trained using stratified Kfold, a method that helped flag problems such as over fitting or selection bias. This scheme has the resistance ability, which is fit for any independent dataset and therefore suitable to be used in predicting new attacks.

6.5 Objective four

The fourth objective was to validate the effectiveness of the prototype. Its capability was tested and excellent performance of Bayesian network was pointed out by four experts in the field of information security who pointed out the most important features to consider when evaluating an IDS. Computational time was noted to be a critical metric used in evaluating effectiveness of an IDS and this study identifies Naïve bayes to have a higher speed of building the model (11.01 seconds) compared to Support Vector Machine which took a longer time (1030.03 seconds).Bayesian network is therefore an ideal solution for real time security analysis especially when the IDS is to be used in real time intrusion response.

CHAPTER 7: CONCLUSION, RECOMMENDATIONS AND FUTURE WORK

7.1 Introduction

This study was focused on development of a scheme that would improve prediction of attacks. The solution was able to predict both normal/good connections and bad connections.

7.2 Conclusion

Existing techniques reviewed for anomaly intrusion detection such as Neural networks, Decision Trees and Bayesian networks helped the researcher settle on Naïve Bayes, a Bayesian based approach. The study presented a strong scheme that was able to differentiate between normal(good) and bad connections(attacks/intrusions).The scheme was able to predict normal class types and various attack types such as 'dos','u2r','r2l' and 'probe'. The results of the predictions have been visualized using graphs in the previous chapter. The scheme has shown consistency and stability in its performance in terms of accuracy, precision and ROC curves as indicated by the performance in the two experiments.

7.3 Recommendations

Bayesian filters can be used in parallel to one another, where an individual filter is designed to detect one type of record. This can improve the accuracy rate of classification and will be useful in attack predictions, an idea that can be exploited by other researchers.

7.4 Future work

In future, a data set to be used can be extended to capture more attack types. Rules used in this scheme could be applied in rule based IDS's such as Snort or any other online intrusion detection system. This can be achieved by using a new data set to obtain rules that would be used on Snort or any other rule based IDS.A classifier that is exposed to more observations, learns and is able to make predictions that are more accurate

References

Ahmad, N., & Habib, M. K. (2017). Analysis of network security threats and vulnerabilities: By development & implementation of a security network monitoring solution.

Alma Cemerlic, Li Yang, Joseph M. Kizza Network Intrusion Detection Based on Bayesian Networks.January 2008

Amudha, P., Karthik, S., & Sivakumari, S. (2013). Classification techniques for intrusion detection an overview. International Journal of Computer Applications, 76(16), 33-40. https://doi.org/10.5120/13334-0928

Bonifacio, J., Cansian, A., De Carvalho, A., & Moreira, E. (n.d.). Neural Networks applied in intrusion detection systems. 1998 IEEE International Joint Conference on Neural Networks Proceedings. IEEE World Congress on Computational Intelligence (Cat.

No.98CH36227). doi:10.1109/ijcnn.1998.682263

Bul'ajoul, W., James, A., & Pannu, M. (2015). Improving network intrusion detection system performance through quality of service configuration and parallel technology. Journal of Computer and System Sciences, 81(6), 981-999. https://doi.org/10.1016/j.jcss.2014.12.012

Cansian et al., 1997),.Retrieved from ProQuest Dissertations and Theses Global.

Chitrakar, R., & Huang, C. (2012). Anomaly based intrusion detection using hybrid learning approach of combining K-medoids clustering and naive Bayes classification. 2012 8th International Conference on Wireless Communications, Networking and Mobile Computing. https://doi.org/10.1109/wicom.2012.6478433

Cunningham, R. K., Lippmann, R. P., Fried, D. J., Garfinkel, S. L., Graf, I., Kendall, K. R., Webster, S. E., Wyschogrod, D., & Zissman, M. A. (1999). Evaluating intrusion detection systems without attacking your friends: The 1998 DARPA intrusion detection evaluation. https://doi.org/10.21236/ada526274

Dayal Ambedkar, M., Ambedkar, N. S., & Raw, R. S. (2016). A comprehensive inspection of cross site scripting attack. 2016 International Conference on Computing, Communication and Automation (ICCCA). doi:10.1109/ccaa.2016.7813770

Debar, H., Becker, M., & Siboni, D. (n.d.). A neural Network component for an intrusion detection system. Proceedings 1992 IEEE Computer Society Symposium on Research in Security and Privacy. doi:10.1109/risp.1992.213257

Demertzis, K., & Iliadis, L. (2014). A hybrid network anomaly and intrusion detection approach based on evolving spiking neural network classification. Communications in Computer and Information Science, 11-23. https://doi.org/10.1007/978-3-319-11710-2_2

Fernandez, A., Garcia, S., Herrera, F., & Chawla, N. V. (2018). SMOTE for learning from Imbalanced data: Progress and challenges, marking the 15-year anniversary. Journal of Artificial Intelligence Research, 61, 863-905. doi:10.1613/jair.1.11192

Frank, E., Hall, M., Holmes, G., Kirkby, R., Pfahringer, B., Witten, I. H., & Trigg, L. (2009). Weka-A machine learning workbench for data mining. Data Mining and Knowledge Discovery Handbook, 1269-1277. doi:

10.1007/978-0-387-09823-4_66

Garthwaite, P. H., Jolliffe, I. T., Professor of Statistics Ian Jolliffe, Jolliffe, I. T., & Jones, B. (2002). Statistical inference. Oxford University Press on Demand.

Gharibian, F., & Ghorbani, A. A. (2007). Comparative study of supervised machine learning techniques for intrusion detection. Fifth Annual Conference on Communication Networks and Services Research (CNSR '07). doi:10.1109/cnsr.2007.22

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The WEKA data mining software. ACM SIGKDD Explorations Newsletter, 11(1), 10-18. doi:10.1145/1656274.1656278

Han, J., Kamber, M., & Pei, J. (2012). Advanced pattern mining. Data Mining, 279-325. doi:10.1016/b978-0-12-381479-1.00007-1

He, S., Zhu, J., He, P., & Lyu, M. R. (2016). Experience report: System log analysis for anomaly detection. 2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE). doi:10.1109/issre.2016.21

Heckerman, D. (1995). Learning with Bayesian Networks. Machine Learning Proceedings 1995, 588. doi:10.1016/b978-1-55860-377-6.50079-7

https://www.researchgate.net/publication/228522321_Anomaly-based_intrusion_detection_in_software_as_a_service [accessed May 17 2021].

https://www.secureworks.com/blog/the-evolution-of-intrusion-detection-prevention

Internet world stats - Usage and population statistics. (n.d.). https://www.internetworldstats.com

Jolliffe, I. (2013). undefined. Springer Science & Business Media.

Kruegel, C., Valeur, F., Vigna, G., & Kemmerer, R. (n.d.). Stateful intrusion detection for high-speed Network's. Proceedings 2002 IEEE Symposium on Security and Privacy. doi:10.1109/secpri.2002.1004378

Kumaravel, Harish Valayapalayam, "An anomaly-based intrusion detection system based on artificial immune system (AIS) [Master's thesis]. (2016). artificialhttps://docs.lib.purdue.edu/open_access_theses/964

Kurose, J. F., & Ross, K. W. (2013). Computer Networking: A top-down approach. Addison-Wesley Longman.

Lapan, S. D., Quartaroli, M. T., & Riemer, F. J. (2012). Qualitative research: An introduction to methods and designs. San Francisco: Jossey-Bass.

Liang, Y., Xu, Q., Li, H., & Cao, D. (2016).Retrieved from ProQuest Dissertations and Theses Global.

Lowe, D. (2005). Networking all-in-One desk reference for dummies. John Wiley & Sons.

M. Mehdi, S. Zair, A. Anou and M. Bensebti, "A Bayesian Networks in Intrusion Detection Systems", Journal of Computer Science 3 (5): 259-265, 2007, ISSN 1549-3636, © 2007 Science Publications.

Mazini, M., Shirazi, B., & Mahdavi, I. (2019). Anomaly Network-based intrusion detection system using a reliable hybrid artificial bee colony and AdaBoost algorithms. Journal of King Saud University - Computer and Information Sciences, 31(4), 541-553. doi:10.1016/j.jksuci.2018.03.011

mirunan999. (n.d.). software 50. Https://Www.Coursehero.Com/. Retrieved 0, from https://www.coursehero.com/

Moayedikia, A., Ong, K., Boo, Y. L., Yeoh, W. G., & Jensen, R. (2017). Feature selection for high dimensional imbalanced class data using harmony search. Engineering Applications of Artificial Intelligence, 57, 38-49. doi:10.1016/j.engappai.2016.10.008

Mukkamala, S., Janoski, G., & Sung, A. (2002). Monitoring system security using neural Networks and support vector machines. Hybrid Information Systems, 121-137. doi:10.1007/978-3-7908-1782-9_10

Myers, G, J Badgett, T, & Sandler, C 2012) Myers, G. J., Badgett, T., & Sandler, C. (2012). A Self-Assessment Test, in The Art of Software Testing. New Jersey: John Wiley & Sons, Inc.

N.A.-M.K.H.-. (2010). Analysis of Network Security Threats and Vulnerabilities by Development & Implementation of a Security Network Monitoring Solution. Analysis of Network Security Threats and Vulnerabilities by Development & Implementation of a Security Network Monitoring Solution. Published.

Naik. (2019). 2019 Issues and Recent Advances in Machine Learning Techniques for Intrusion Detection Systems. naik.

Nieles, M., Dempsey, K., & Pillitteri, V. Y. (2017). An introduction to information security. An Introduction to Information Security. Published. https://doi.org/10.6028/nist.sp.800-12r1

Ohta S, R. Kurebayashi and K. Kobayashi. , Minimizing false positives of a decision tree classifier for intrusion detection on the internet, Journal of Networks System Management, vol.16, 2008, pp.399–419. ISSN 1064- 7570.

Panda, M., Abraham, A., Das, S., & Patra, M. R. (2011). Network intrusion detection system: A machine learning approach. Intelligent Decision Technologies, 5(4), 347-356. https://doi.org/10.3233/idt-2011-0117

Paquet, C. (2009). Implementing Cisco IOS Network security (IINS): (CCNA security exam 640-553) (Authorized self-study guide). Cisco Press.

Practical Automated Detection of Stealthy Portscans. (2002). Practical Automated Detection of Stealthy Portscans. Published. https://doi.org/10.3233/JCS-2002-101-205

Ramadas, M., Ostermann, S., & Tjaden, B. (2003). Detecting anomalous Network traffic with self-organizing maps. Lecture Notes in Computer Science, 36-54. doi:10.1007/978-3-540-45248-5_3

Reid, S. (2005). The art of software testing, second edition. Glenford J. Myers. Revised and updated by Tom Badgett and Todd M. Thomas, with Corey Sandler. John Wiley and sons, New Jersey, U.S.A., 2004. ISBN: 0-471-46912-2, pp 234. Software Testing, Verification and Reliability, 15(2), 136-137. doi:10.1002/stvr.322

Sabhnani, M., & Serpen, G. (2004). Why machine learning algorithms fail in misuse detection on KDD intrusion detection data set. Intelligent Data Analysis, 8(4), 403-415. doi:10.3233/ida-2004-8406

Salman, T., Bhamare, D., Erbad, A., Jain, R., & Samaka, M. (2017). Machine learning for anomaly detection and categorization in multi-cloud environments. 2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud). https://doi.org/10.1109/cscloud.2017.15

Shin, S., Kwon, T., Jo, G., Park, Y., & Rhy, H. (2010). undefined. IEEE Transactions on Industrial Informatics, 6(4), 744-757. https://doi.org/10.1109/tii.2010.2051556

Silverstone, 2003.Retrieved from ProQuest Dissertations and Theses Global.

The main benefits of computer networking in 2017. (2017, February 3). Inspired Techs. https://www.inspiredtechs.com.au/computer-networking

Thomas, C. (2020). Introductory chapter: Computer security threats. Computer Security Threats. https://doi.org/10.5772/intechopen.93041

Us-en_cloud_learnhub_networking-a-complete-guide. (2019, June 25). IBM - United States. https://www.ibm.com/cloud/learn/networking-a-complete-guide

Wang and S.J. Stolfo. Anomalous payload-based networkintrusion detection. InProceedings of the 7th InternationalConference on Recent Advances in Intrusion Detection, pages203–222. Springer, 2004

Wang, Y. (2009). Statistical techniques for Network security.Retrieved from ProQuest Dissertations and Theses Global.

Weaver, R., Weaver, D., & Farwood, D. (2013). Guide to network defense and countermeasures. Cengage Learning.

Weaver, R., Weaver, D., & Farwood, D. (2013). Guide to network defense and countermeasures. Cengage Learning.

Weiming Hu, Wei Hu, & Maybank, S. (2008). Adaboost-based algorithm for Network intrusion detection. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 38(2), 577-583. doi:10.1109/tsmcb.2007.91469

Appendix A

Plagiarism Similarity Report

Curiginal

Document Information

Analyzed document	Dissertation.pdf (D111131096)
Submitted	8/10/2021 10:36:00 AM
Submitted by	
Submitter email	pamela.kirui@strathmore.edu
Similarity	8%
Analysis address	library.strath@analysis.urkund.com

Sources included in the report

W	URL: http://repo.uofg.edu.sd/bitstream/handle/123456789/1412/Osama%20Abdulmonim%20Musa%20Mohamed%20.pdf?sequence=1&isAllowed=y Fetched: 1/13/2021 5:25:09 PM	2
W	URL: http://cse.anits.edu.in/projects/projects1920C12.pdf Fetched: 7/19/2021 6:52:16 AM	2
SA	**final report.docx** Document final report.docx (D22943790)	1
SA	**Chapter1-2-3-4-5-6(Reference_Complete) 12 June 2016.docx** Document Chapter1-2-3-4-5-6(Reference_Complete) 12 June 2016.docx (D25539915)	1
W	URL: https://www.ibm.com/cloud/learn/networking-a-complete-guide Fetched: 8/10/2021 10:36:00 AM	2
W	URL: https://www.inspiredtechs.com.au/computer-networking Fetched: 8/10/2021 10:36:00 AM	1
J	**HYBRID NETWORK BASED INTRUSION DETECTION SYSTEMS FOR DoS ATTACKS** URL: 8b883665-effa-4426-a50e-f9e9c111137c Fetched: 3/16/2019 2:23:41 PM	1
SA	**neha final till 18 sep urkund.docx** Document neha final till 18 sep urkund.docx (D30610379)	1
W	URL: https://pdfs.semanticscholar.org/3d8f/755c22ba7d1cee7cc2f93decd651816d5736.pdf Fetched: 11/29/2020 8:23:05 AM	6
SA	**Sofia. P.docx** Document Sofia. P.docx (D32741123)	2
W	URL: http://www.sbsstc.ac.in/icccs2014/Papers/Paper45.pdf Fetched: 5/3/2021 11:42:27 AM	2
SA	**complete report shubh.doc** Document complete report shubh.doc (D33939351)	2

Curiginal

W	URL: https://www.researchgate.net/publication/224710672_Comparative_Study_of_Supervised_Machine_Learning_Techniques_for_Intrusion_Detection Fetched: 4/8/2020 8:57:52 AM	2
SA	**Senthinayaki_Thesis.docx** Document Senthinayaki_Thesis.docx (D18565286)	1
SA	**Anmol Saxena_Major_Project_II_report.docx** Document Anmol Saxena_Major_Project_II_report.docx (D38138462)	2
SA	**A Study on NSL Dataset for Intrusion Detection System based on Classification Algorithms.docx** Document A Study on NSL Dataset for Intrusion Detection System based on Classification Algorithms.docx (D18736062)	1
W	URL: http://www.ijaiem.org/Volume2Issue6/IJAIEM-2013-06-09-029.pdf Fetched: 8/6/2021 7:58:59 AM	1
W	URL: https://doi.org/10.1109/wicom.2012.6478433 Fetched: 8/10/2021 10:36:00 AM	1
W	URL: https://repository.ubuntunet.net/bitstream/handle/10.20374/237/zvareshek.pdf?sequence=1&isAllowed=y Fetched: 11/20/2019 7:03:25 AM	2
SA	**SR Venugopalan - PhD Thesis - Computer Science.pdf** Document SR Venugopalan - PhD Thesis - Computer Science.pdf (D32047764)	2
W	URL: https://docs.lib.purdue.edu/open_access_theses/964 Fetched: 8/10/2021 10:36:00 AM	1
J	**Implementation and Analysis of Combined Machine Learning Method for Intrusion Detection System.(Report)** URL: 253564b6-ed26-44d2-93c7-47359731766f Fetched: 3/9/2019 4:17:09 AM	1
W	URL: https://www.iieta.org/journals/ria/paper/10.18280/ria.340310 Fetched: 7/27/2021 5:57:55 PM	1