IMPROVING INTRUSION DETECTION SYSTEMS USING DATA MINING TECHNIQUES

By

Abdulrazaq Almutairi

A Doctoral Thesis submitted in partial fulfilment of the requirements for the award of Doctor of Philosophy of

Loughborough University

9th February 2016

ABSTRACT

Recent surveys and studies have shown that cyber-attacks have caused a lot of damage to organisations, governments, and individuals around the world. Although developments are constantly occurring in the computer security field, cyber-attacks still cause damage as they are developed and evolved by hackers. This research looked at some industrial challenges in the intrusion detection area. The research identified two main challenges; the first one is that signature-based intrusion detection systems such as SNORT lack the capability of detecting attacks with new signatures without human intervention. The other challenge is related to multi-stage attack detection, it has been found that signature-based is not efficient in this area. The novelty in this research is presented through developing methodologies tackling the mentioned challenges. The first challenge was handled by developing a multi-layer classification methodology. The first layer is based on decision tree, while the second layer is a hybrid module that uses two data mining techniques; neural network, and fuzzy logic. The second layer will try to detect new attacks in case the first one fails to detect. This system detects attacks with new signatures, and then updates the SNORT signature holder automatically, without any human intervention. The obtained results have shown that a high detection rate has been obtained with attacks having new signatures. However, it has been found that the false positive rate needs to be lowered. The second challenge was approached by evaluating IP information using fuzzy logic. This approach looks at the identity of participants in the traffic, rather than the sequence and contents of the traffic. The results have shown that this approach can help in predicting attacks at very early stages in some scenarios. However, it has been found that combining this approach with a different approach that looks at the sequence and contents of the traffic, such as event- correlation, will achieve a better performance than each approach individually.

KEYWORDS

Intrusion detection; Multi-Stage attacks; IP Check; Data mining; Fuzzy Logic; Neural Network; Naïve base; Decision tree; SNORT

ACKNOWLEDGEMENTS

First I would like to thank God who has been a source of blessing and making this project come through.

I would like this opportunity to thank my supervisors, Prof. David Parish and Dr. James Flint who have been very patient with me and have been relentlessly guiding me with their expert guidance.

I would also like to thank all of my colleagues in High Speed Network group for their support, help and for sharing their knowledge, Dr. Francisco Aparicio-Navarro, Dr. Konstantinos Kyriakopoulos and Gines Escudero-Andreu.

I would also like to thank family especially my wife Dr. Shaimaa Almutairi and friends who have helped with number of tasks in the PHd. I would also like to extend my gratitude to my sister – Huda, for whom I wish that she will wake up from coma and read this thesis. And I would like to thank my parents for their consistent support, without them I would not be where I am today.

Finally I would also thank my country Kuwait. And I would like to thank Prime Minister of the State of Kuwait Sheikh Jaber Mubarak Al-Hamad Al-Sabah, And Sheikh Mohammad Abdullah Al-Mubarak Al-Sabah, and Sheikh Khaled Mohammad al-Khaled Al-Sabah.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

GLOSSARY OF TERMS

- **AD:** Anomaly-based Detection.
- **Anonymous Proxy:** It is a server that acts as a middleman between a machine and the internet. It is used to hide a user's identity when communicating with the internet.
- **Bot:** is a program that is designed to carry out a number of tasks and normally waits for orders to be executed from a master computer.
- **Botnet:** a group of machines that are infected by a bot controlled by the same master.
- **CFS:** correlation-based feature selection.
- **Confusion matrix:** representation of actual and predicted results for a classification. In this context, it is used to measure the performance of a classification system.
- **CPU:** Computer Processing Unit.
- **CRLF:** Carriage Return Line Feed.
- **Cross Site scripting:** it is a vulnerability that allows attackers to inject client side code to a web page accessed by other users.
- **DDOS attacks:** is a distributed denial of service. It is a denial of service that is carried out in a planned manner by multiple attackers targeting one victim.
- **Denial of Service:** a denial of service is defined as an attempt by attackers to affect a machine in a way that it will not deliver a service for users having permissions of access. Affected machines become irresponsive in this case.
- **DNS:** Domain Name System.
- **DNSBL:** Domain Name System Block Lists.
- **False negative:** In this context, false negative refers to alert not raised by a system while it is supposed to be raised. In this case, the system thinks it is a normal traffic while it is an attack.
- **False positive:** In this context, false positive refers to alert raised by a system while it is not supposed to be raised. In this case, the system thinks it is an attack while it is normal traffic.
- **FCBF:** Fast Correlation Based Filter.

- **Firewall**: It is a software or hardware system designed to check incoming traffic from outside the network (e.g. the Internet), and then decides either to pass the traffic or stop it based on the firewall settings (e.g. block unauthorized access).
- **Flood attacks:** can be defined as any kind of attack that is carried out to target a system by overwhelming the system resources. It can be achieved by flooding the system with a large number of requests or responses.
- **FTTP:** File Transfer Protocol.
- **GR:** gain ratio.
- **Honeypot:** Bandy (2015) defined it as a tool used to protect networks from unauthorized access, it does not contain data or applications that are critical to an organization but it has some data that hacker have an interest in. In other word, it is a computer in a network configured to interact with hackers in order to get some details about their attacks.
- **HTTP:** Hyper Text Transfer Protocol.
- **ICMP:** Internet Control Messaging Protocol.
- **IG:** information gain.
- **Intrusion Detection Systems:** An Intrusion Detection System is a software or hardware system that monitors incoming and outgoing network traffic and raises an alert when detecting malicious activities in the traffic. More details about intrusion detection systems are included in chapter two.
- **Intrusion Prevention System:** it is an Intrusion Detection System but able to take an action when detecting malicious traffic.
- **ISP:** Internet Service Provider.
- **KDD:** Knowledge Discovery and Data.
- **LAMP:** it is a development framework that includes Linux as an operating system, APACHE as a web server, MYSQL as a database, and PHP as a programming language.
- **LARIAT:** Lincoln Adaptable Real-time Information Assurance Testbed.
- **MACE:** Malicious Traffic Composition Environment.
- **MLP:** Multi-Level Perceptron.
- **Multi stage attack:** It is an attack that occurs through multiple steps without violating any rules. More details about this type of attacks are discussed in chapter four.

- **NMAP:** Network Mapper.
- **NNTP:** Network News Transfer Protocol.
- **PCAP:** Packet Capture.
- **RAM:** Random Access Memory.
- **SD:** Signature-based Detection.
- **SNORT:** is a signature based intrusion detection system.
- **SPA:** Stateful Protocol Analysis.
- **SPAMS:** this expression is used when referring to sending a large amount of unrequested emails.
- **Threats:** the possibility of exploiting a vulnerability to carry out an attack targeting the system having the vulnerabilities.
- **Trace file:** a file that contains activities belonging to a user or software. In this context, trace file contains network activities (outgoing and incoming packets).
- **True negative:** In this context, true negative refers to the correct behaviour of a system when there is no attack. The system does not raise any alert in this case.
- **True positive:** In this context, true positive refers to the correct behaviour of a system when there is an attack. The system raises any alert in this case.
- **URL:** Uniform Resource Locator
- **Virus:** a program that is developed with a malicious purpose to affect a system in a harmful way.
- **Vulnerabilities:** A vulnerability can be defined as a weakness in a system design, implementation, or configuration that can be exploited by an attacker resulting in security breach, overcoming the system's security policy, or leading to compromising a machine.
- **Worms:** a worm is malicious software that can spread itself across networks without the need of any human intervention through emails and file sharing etc.
- **XSF:** Cross Site Forgery.
- **XSS:** Cross Site Scripting.
- **Zombie army:** is a machine connected to the Internet configured to forward malicious traffic (including spam or viruses) to other machines on the Internet, without any permission from the machine owner.

- **Zombies:** A machine infected with a malicious program and set to be a part of a botnet.
- **NIDS:** Network Based Intrusion Detection System.
- **HIDS:** Host Based Intrusion Detection Systems.

CHAPTER 1

INTRODUCTION

1.1 Background

There is no doubt that the Internet plays an important role in different aspects of life these days. For example, it has been found that social networking such as Facebook, Twitter, and Linked-in have a remarkable impact in bringing people from different parts of the world together (Muila, 2010). Although it has changed the world, it has raised the possibility that malicious users gain illegal access to organizations to steal confidential information they are interested in or destroy it by injecting applications called malware. Those applications are created to give malicious users the ability to control organizations' computers remotely. Malicious users get an illegal access to those organizations by exploiting weaknesses and vulnerabilities in organizations' networks or web applications. The impact of attacks can lead to delaying delivering services in some organizations causing financial damages. A survey made by Statistia (2015) provides information on the distribution of costs for external consequences of targeted cyber-attacks on companies in global markets in 2014. Figure 1.1 shows the results obtained in that survey, it was found that 38 percent of participants pointed to business disruption as the most expensive consequence of a cyber-attack on their business.

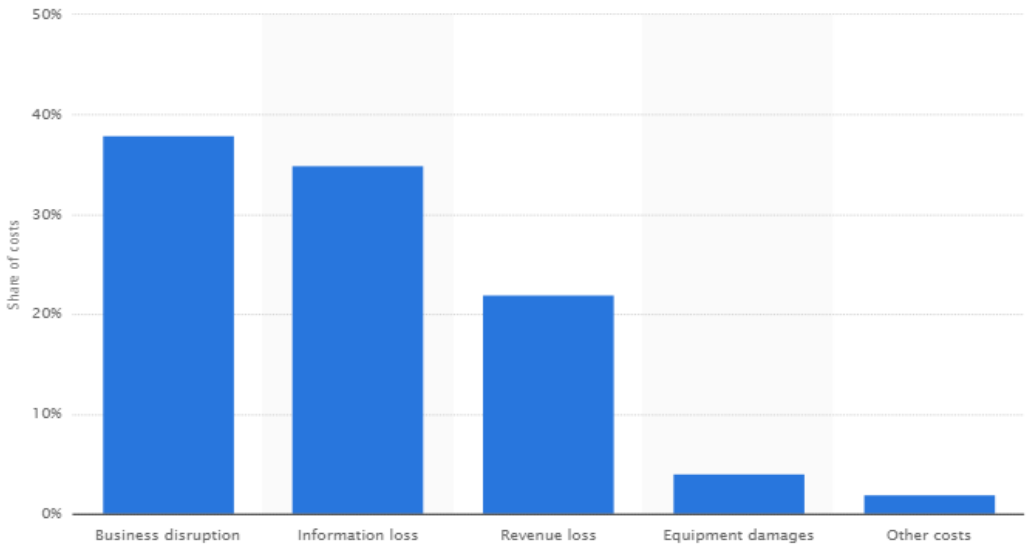

Figure 1.1: Distribution of costs for external consequences of targeted cyber-attacks reported by Statistia (2015)

Williams (2014) reported that cyber-attacks were estimated to cost the global economy around $445 billion annually. She also reported that those attacks affected more than 800 million people in 2013. An annual study conducted by the Ponemon institute (2014) in seven countries including the United States, United Kingdom, Germany, Australia, Japan, France and the Russian Federation. The study involves a total benchmark sample of 257 organizations. Figure 1.2 presents the estimated average cost of cyber attacks for each country, it has been found that the US sample achieved the highest total average cost at $12.7 million while the Russian Federation sample got the lowest total average cost at $3.3 million. The figure also that the cost of cyber attacks went up in six countries during the past year compared to 2013 (apart of the Russian Federation), the highest increase was found in the United Kingdom (22.7%) while the lowest increase was found in Japan (2.7%). The study also reported that all industries are targeted by cyber-attacks, but with different levels. The study pointed out that organisation providing energy and financial services experience higher cyber-attack costs than organizations providing services in media, life sciences and healthcare.

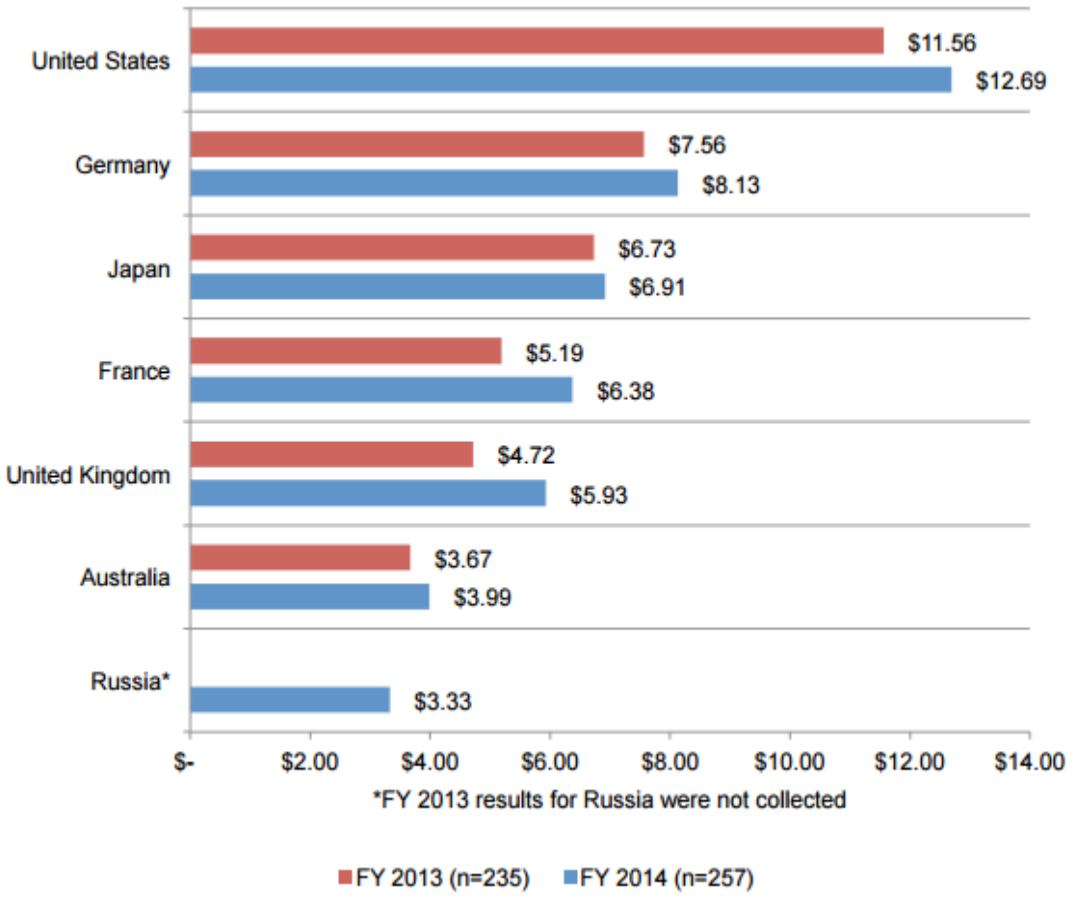

Figure 1.2: Distribution of costs for external consequences of targeted cyber-attacks reported by Ponemon institute (2014)

An example of those attacks is a cyber-attack targeted the Dutch government's main website for most of 10th February 2015, it was reported by Reuters (2015) about that incident the following:

"Cyber attackers crippled the Dutch government's main websites for most of Tuesday and back-up plans proved ineffective, exposing the vulnerability of critical infrastructure at a time of heightened concern about online security. The outage affected most of the central government's major websites, which provide information to the public and the media"

Another example of those attacks was against the online payment site PayPal in 2010. Rawlings (2013) reported that after WikiLeaks had issued a lot of classified material, PayPal decided to block WikiLeaks' accounts in a way that stop anti-secrecy site from receiving online donations. That action pushed the anonymous group to launch an attack against PayPal, the aim of their efforts was to make the access to the

website impossible affecting all integrations between many of websites and paypal. The cost of this attack was estimated to be £3.5 million.

The impact of such attacks has made the internet security not only a matter related to businesses and organizations but extended to include national pushing governments to play an important role in that area (Statistia, 2015).

Detecting malicious activities occurring in a computers or networks can be achieved using IDS which is considered a security management system that monitor network traffic and raise an alert when capturing malicious activities. IDS have been widely employed in many organizations to detect attacks. The increase of their usage is down to availability of IDS as free of charge, and open source. In addition, there is a wide community of exports (Muila, 2010). Although the wide spread and usage of IDS over the world, there are many challenges that make detecting some attacks difficult. One of the main challenges is the ability to provide protection against new attacks. Cyber attacks can get more costly for an organization if not detected quickly (Ponemon, 2014).That challenge was described by SANS (2001) as following:

"The IDS technology is still reactive rather than proactive. The IDS technology works on attack signatures. Attack signatures are attack patterns of previous attacks. The signature database needs to be updated whenever a different kind of attack is detected and the fix for the same is available. The frequency of signature update varies from vendor to vendor."

Another challenge is minimizing human intervention. Werlinger (2008) reported that IDS require a lot of human resources in the monitoring and analysis phases to investigate captured attacks and tune the system in order to reduce number of false alarms. Therefore, managing to minimize human intervention will lead to minimizing the operational cost of using IDS inside an organization.

One of IDS tools that is commonly used and has many researchers conducted to improve it is SNORT It is an example of signature based IDS. Many researches were conducted to evaluate the performance of this tool. There are also many efforts made by researchers to improve SNORT detection capabilities using data mining techniques such as genetic algorithm, and decision tree. In this research, a multi-layer system based on data mining techniques is proposed to update automatically the SNORT's (an open source network intrusion prevention system) signature holder

without the need of any human intervention. In addition, part of the solution proposed in this research also work in conjunction with SNORT to detect multi stage attacks that SNORT has limited capabilities to detect them.

1.2 Motivations of the research

The industrial challenges for improving intrusion detection systems are the motivation of this research project. As mentioned in the background section, minimizing the human intervention is one of those challenges. Although intrusion detection systems manage to reduce the time spent in capturing suspicious activities, other actions have been found dependant on human interventions. One of those actions is controlling the sensitivity to reduce the false positive rate. In addition, human intervention is required to update some intrusion detection systems with new rules to detect new attacks. Such actions are very time consuming, automating some of those actions will speed up the process of identifying intrusions and consequently will lead to a drop in the cost of attack response cycle (Hawrylkiw, 2002).

Detecting or predicting multi stage attacks is another challenge that is worth considering. Multistage attacks can evolve dramatically these days, causing much loss and damage to organisations. These attacks occur through multiple steps, each step looking legal and not violating any rules for some intrusion detection systems. Different solutions have been introduced to detect multi-stage attacks, some of those being event correlation-based. Event-correlation based solutions try to match network events with certain attack patterns. When a stream of network events matches a certain pattern, attacks can be stopped before progressing to the next stages. Many researchers claim the effectiveness of that approach in detecting multi-stage attacks However, this approach requires having up-to-date multi-stage attack patterns (sequences), which is not easy to achieve in a very short time, as discovering new complex attacks normally takes some time. The Shady Rat Operation attack is a good example of that; it started in 2006 and was only discovered in 2011 (Tal Global, 2011).

This thesis describes a solution that contributes in overcoming the mentioned challenges. The proposed solution has handled the first challenge by creating an intelligent system integrated with SNORT, this system uses data mining techniques to

detect new attacks not captured by SNORT then updates SNORT with the signature of those attacks automatically. The other challenge has been handled in this research by creating a system that follows IP information evaluation approach. This approach looks at the identity of the network traffic source rather than the sequence. In other words, it asks this question “who is communicating with us?” Rather than “what is being done in our environment?” The attackers usually try to hide their identities by using anonymous proxies. In addition, their traffic in many scenarios involves communications with IPs having bad reputation. Therefore, evaluating IP information (e.g. is the IP an anonymous proxy) can help in predicting potential attacks before their occurrence.

1.3 Research Aim and objectives

This research work aims to improve intrusion detection by proposing a new approach that will work in conjunction with SNORT; this approach will handle some of SNORT shortcomings. One of those shortcomings is the ability to detect recent attacks, the solution will be built in to detect those attacks then update SNORT with signatures of those attacks. Another shortcoming the proposed system will handle is its deficiency in detecting some multi stage attack scenarios; the other solution will not interact with SNORT. The proposed solution uses several data mining techniques in handling those shortcomings. In order to achieve the aim (using the approach), the following objectives would be met:

- Conducting a literature survey about intrusion detection systems by looking at different IDS tools and researches carried out using data mining techniques to improve them.
- Finding suitable data set for training and evaluation as the solution will use some machine learning algorithms.
- Building a classifier (first layer) based on machine learning algorithm that will be considered as the first defence line against new attacks.
- Building a reasoning module that will act as a second layer of classification module for traffic that the first layer will fail to classify.
- Analysing four different multi stage attack scenarios to understand multi stage attack behaviour.

- Building a solution that predicts multi stage attacks based on the analysis carried out on multi stage attack scenarios.
- Measuring the effectiveness of the system that detects multi stage attacks using metrics based approach.

1.4 Thesis original contribution

The contributions made in the thesis are as follows:

1. A methodology has been proposed that aims to improve the SNORT performance by automating adding the signature of recent attacks. The methodology involves two layers. The first layer is decision tree based while the second layers is a hybrid module that uses a neural network and the fuzzy logic. Using three different data mining techniques over the two layers reduces the chance of passing new attacks without detection.
2. A methodology has been introduced to predict/detect multi stage attacks based on evaluating IP information using fuzzy log. The methodology involves using of three modules; network sniffing, IP information finder, and the reasoning module.
3. A validation approach has been used to evaluate the approach used to detect multi stage attacks. The validation approach is a modified version of a metrics-based approach introduced in a validation study.

1.5 Research Methodology

The research methodology followed in this research is a combination between qualitative and iterative experimental approach. The qualitative approach has been used to understand some concepts and systems behaviour while the iterative experimental approach is used when building the systems. In the iterative experimental approach, the system is initial implemented then tested. Based on the obtained results, the system implementation is modified until reaching to a point where obtained results are acceptable.

1.6 Thesis Outline

- **Chapter 2:** The second chapter in this thesis provides a literature review. It provides a discussion various types of honeypots, the discussion includes detailing monitoring methods used by Honeypots. In addition, it enumerates advantages and shortcomings of Honeypots/Honeynet. The chapter also goes through IDS giving a quick overview then going through different intrusion detection methodologies, available IDS tools in the market detailing their cons and pros, and talking about limitations of IDS. This part of the thesis also walks through different data mining techniques and how they are employed in improving intrusion detection systems by some researchers and result obtained by that employment. Moreover, the chapter looks at feature selection algorithms showing how it can play a vital role in IDS.
- **Chapter 3:** The third chapter in this research proposes a system that can be used to improve SNORT. The proposed system uses data mining techniques to detect malicious packets that SNORT is not able to capture then automatically updates SNORT signatures holder with a new one. The chapter provides a brief background overview over data mining techniques used in the proposed system (Decision tree, Naïve Base, Neural Network, Fuzzy logic). The chapter goes through the data set used for training and evaluation (KDD99) and which features from the data are selected in the proposed system. In addition, it shows how each module is trained and evaluated. This part of the thesis provides the evaluation results obtained from the proposed system in a form a confusion matrix showing how well the system is capable of performing its job.
- **Chapter 4:** This chapter goes through four different multi-stage attack scenarios. The aim of this chapter is to understand the behaviour of multi-stage attacks and try to find a clue to predicting or detecting such kinds of attacks. In each scenario, the network traffic will be analysed highlighting all steps that have occurred and not been considered by many security systems. The outcome from analysing each scenario will be in the form of rules that will be used in building a solution that will predict multi-stage attacks before they have an impact and damage organisations.

- **Chapter 5:** This chapter presents the proposed solution to detect multi stage attacks. It goes through the architecture of the solution detailing the interaction between different modules. In addition to this, it goes through each module individually showing different options for implementation and reasoning why one of them is preferred over others. The chapter includes some pieces of the codes that show the logic in each module.
- **Chapter 6:** This chapter presents the evaluation for the solution proposed in chapter 5. The evaluation process involves following a metrics based approach then comparing the proposed solution with solutions proposed by other researchers. The metrics based approach evaluates the system from different perspectives; it includes logistics metrics, design metrics, and performance metrics.
- **Chapter 7:** This chapter presents the conclusion and future work. It provides a critical review of the work done and pointing to areas than require some enhancement to achieve better results.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction and Background

Information security is becoming a primary concern in this age of information. The classical method of security that was more or less defensive is now being scaled to more aggressive defence format. Intrusion detection is a process of monitoring networks and machine within the network for unauthorised usage and or activity. Meijerink and Spellen (2006) define Intrusion Detection Systems (IDSs) as a system for detection of unwanted manipulation to system. This manipulation may be in form of attack by an attacker or simply by use of malicious script that changes the fingerprint of the system under attack (Kuwatly et al., 2004). Typically, an IDS is required for detection of all malicious traffic that cannot be detected by generally deployed tools such as firewall. IDSs are categorised into host-based (HIDS) – where data on individual computer system is examined and network-based (NIDS) – where analysis of data packets that transit over the network is carried out. According to Meijerink and Spellen (2006), various types of network attacks include data driven attacks on applications and

network attack on services and host attacks include unauthorised logins, setup of malware, access to files and changing of privileges.

Unlike common IDSs, honeypot technology tends to provide the attacker with important resources that are needed for a successful attack. Honeypot or honeynet based decoy system is implemented for the purpose of intrusion detection and protection. A honeypot is difficult to define as there are number of interpretations that have been understood from the literature which include domains of attack prevention, attack detection, data collection in context of security. It is distinctive as it is technology and not a solution or procedure / process to resolve a particular security issues. A honeypot is a trap set to detect, deflect or in some cases counteract the attempts of unauthorised usage of production systems. It appears to be part of a network but remains isolated and protected. Its value lies in being probed, attacked and compromised (Spitzner, 2003). Hence, honeypots has no production value and they should not work with any legitimate traffic or events. According to (Mokube and Adams, 2007), the purpose of honeypot or monitored honeynet networks include the following:

1. They form a defensive distraction system in order to direct an attacker towards machines containing no valuable information;
2. They serve the purpose of a early warning system that can inform about exploitation trends; and
3. They become a data collection store that can be used to examine the methods and processes of exploitation of a honeypot.

The interaction with honeypots is expected from attackers; hence the value of honeypot lies in unauthorised interaction conducted by abusers of the vulnerable honeypot.

Another term that is used regularly with honeypots is Honeynet that means a network that is formed of one or more honeypots (Gupta et al., 2012). Honeypots are classified based on level of interaction and purpose. Further details regarding the same are discussed in Section 2.

This chapter has been organised as follows: In Section 2, we discuss various types of honeypots. Section 3 goes through purpose of Honeypot followed by three sections (4, 5, and 6) where details regarding monitoring methods used by Honeypots are discussed in detail. Section 7 enumerates advantages and shortcomings of Honeypots / Honeynet. Section 8 discusses IDS giving a quick overview then going through different intrusion detection methodologies and finally talking about limitations of IDS. In section 9, data mining is briefly explained then some of techniques used in data mining are critically discussed. Section 10 talk about feature selection algorithm and how it can play a vital role in IDS. Section 11 provides a discussion about the chapter. Finally, Section 12 gives a summary of this chapter.

2.2 Classification of Honeypots

2.2.1 Low-Interaction Honeypots

A low-interaction honeypot simulates only limited services that cannot be exploited enough to gain total control of the honeypot (Sharma and Sran, 2011). The low level honeypot provides emulating services and operating system to the attacker, which makes it easier to deploy, and maintain. Example of emulated services include FTP service, listening on port 21 (Telnet), login to FTP server etc. The emulated services mitigate risk by containing the attacker's activity. The interaction between this type of honeypot and production system is very limited. These type of honeypots can be compared to passive IDS as network traffic is not modified in any way and they do not interact with the attacker thus mitigating the risk associated with this category of honeypots (Mokube and Adams, 2007). Generally, low-interaction honeypots are used to analyse spammers and can also be used for providing countermeasures against worms.

A well-known example of a commercial low interaction honeypot is Honeyd (Provos, 2003). Honeyd (Provos, 2003) is a daemon that can

used to simulate a large network on a single network host. It is a framework for creating virtual honeypots using unused IP addresses of a network, which simulates various operating systems and services. Other low-interaction honeypot include Specter (Netsec, 2012) and KFSensor (KFSensor, 2012). Specter can monitor a total of 14 TCP ports and of these 14 monitored ports, 7 ports are called traps, and the other 7 are called services. Traps are port listeners: when the attacker makes a connection, the attempt is terminated, and then logged. Services are more advanced where there is interaction with the attacker, emulating the application (Netsec, 2012). The level of emulation depends on each service. For example, the HTTP service emulates a simple Web server with default static Web pages. Figure 2.1, shows the control panel for low-interaction honeypot tool – Specter and KFSensor simulates system services at the application layer (Kuwatly et al., 2004). Reference (KFSensor, 2012) explains the methods in which KFSensor can be used to setup new firewall rules. Figure 2.2, shows the HTTP service emulation within low interaction honeypot tool – KFSensor.

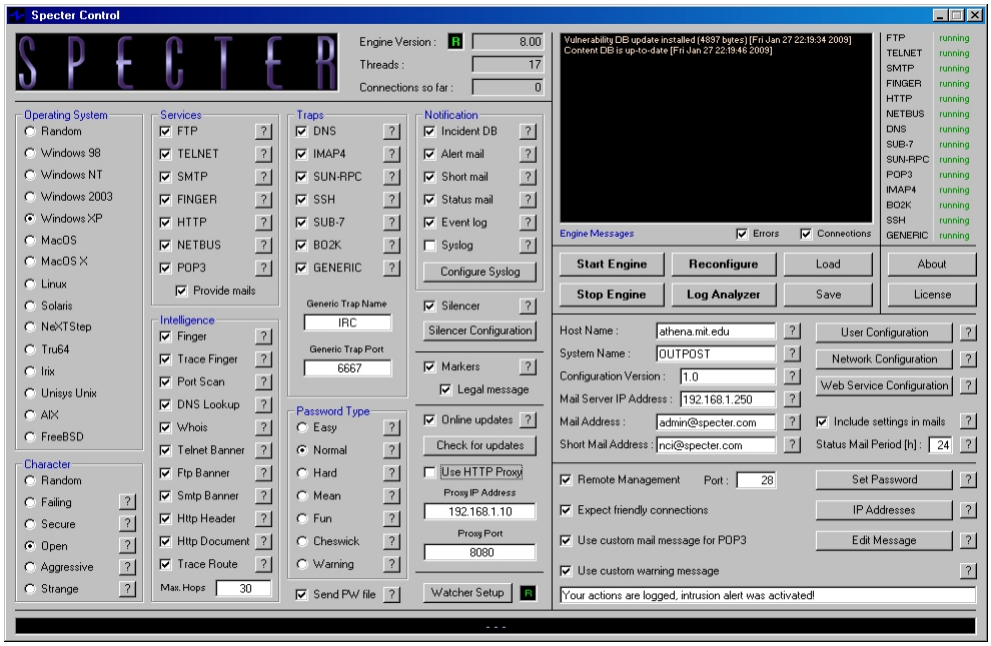

Figure 2.1: Control panel for specter tool showing services that may be emulated (Source: (Netsec, 2012))

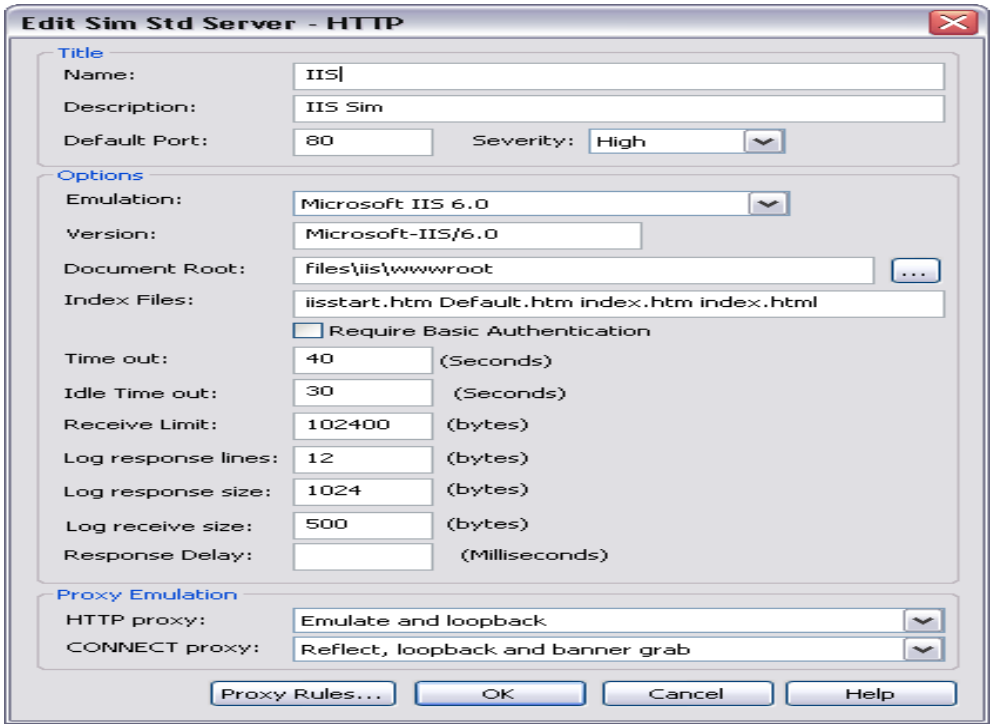

Figure 2.2: HTTP service emulation setup using KFSensor (Source: (KFSensor, 2012))

2.2.2 High-Interaction Honeypots

High interaction honeypots are complex solutions, which include deploying of a real operating systems and applications (Saini et al., 2011). As it involves real operating system, the level of risk is increased by many folds, but it is a trade-off in order to capture extensive amounts of information by allowing the attackers to interact with real systems (Singh and Joshi, 2011). This facilitates capturing / logging of full extent of attacker's behaviour that can be analysed at later stage. According to (Singh and Joshi, 2011), as the attacker has more resources to exploit at his disposal, a high interaction honeypot should be regularly monitored to ensure that it does not become a security issue.

Example of high interaction honeypots include Honeynets (Project, 2012), Sebek (Huang et al., 2009), Argos (Portokalidis et al., 2006). Argos offer a full operating system to the attacker and when the attackers tries to

do something malicious the honeypot will shut down and makes dumps of memory and disk to get information about what the attacker was trying to do (Meijerink and Spellen, 2006). A greater detail regarding the high interaction monitoring methods is discussed in Section 6.

2.3 Purpose of Honeypot

2.3.1 Research Honeypot

A research honeypot is used to gain the information about the attacker's community and does not add any direct value to the organisation (Sadasivam et al., 2005). The purpose of research honeypots is to gather intelligence regarding general threats that an organisation may face and hence allow organisation to protect itself in a better form against those analysed threats. The primary function is to study the method how attacker attacks, understand their objectives and behaviour (Saini et al., 2011). These type of honeypots are like high-interaction honeypots that are complex to deploy and difficult to maintain. They are generally used within research and commercial community in addition to military and defence organisations. According to (Mokube and Adams, 2007), they add tremendous value to research providing a platform to study cyber threats and attacks. They may also be suitable for aiding in development of analysis and forensic skills. (Spitzner, 2000) provides the instance where honeypot was used as a forensic analysis for domain name system (DNS) attack.

2.3.2 Production Honeypots

Production honeypots are used within the environment of a organisation to protect the information assets of the organisation and help in mitigation of risk (Sadasivam et al., 2005). Unlike research honeypots, they have direct values as they provide security to organisation's production resources. As they do not require a large amount of functionality, they are not too complex to deploy or maintain and consequently, they are unable to

provide a large amount of information regarding the attackers. Their primary function is to mirror the production network of the organisation and invite attackers to interact with them, so that vulnerabilities of the network can be exposed. They are considered to add value to detection of attacks rather than prevention of attacks. One the examples of production honeypot is Nepenthes (Baecher et al., 2006).

Classification of Honeypot	Categories of Honeypot	Examples	Brief Description
Level of Interaction	Low Interaction Honeypot	HoneyD, Specter, KFSensor, MWCollect	A low-interaction honeypot simulates only limited services that cannot be exploited enough to gain total control of the honeypot.
	High Interaction Honeypot	Honeynet, Sebek, Argos	High interaction honeypots are complex solutions, which include deploying of real operating systems and applications.
Purpose of Honeypot	Research Honeypot	Honeynets	A research honeypot is used to gain the information about the attacker's community and does not add any direct value to the organisation.
	Production Honeypot	Nepenthes	Production honeypots are used within the environment of an organisation to protect the information assets of the organisation and help in mitigation of risk.

Table 2.1: Classification of honeypots

2.4 Monitoring Methods of Honeypots

Honeypot monitoring is a very important component of any honeypot deployment. There are two methods that used for monitoring of honeypots viz. external method (network-based) and internal method (host-based). In the network-based method, all packets that are sent to or received from the monitored honeypot are captured and traffic sniffing tools such as TCPDUMP (TCPDUMP, 2012) and Ethereal (Ethereal, 2012). In the host-based method, specialised sensors are deployed within the honeypot in order to monitor and record system events.

It should be noted that both approaches have their strengths and weakness. For instance, the network-based approach though being transparent and invisible to the attacker can sniff packets by being deployed outside the honeypot but it cannot capture internal system events on a vulnerable honeypot. Furthermore, it may be ineffective or perform at lower effectiveness, if the network data traffic is encrypted. On the other hand, the host-based method, if detected by the attacker can be tampered with, thus leaving it ineffective.

Data capture modules in high interaction honeypots deals with collection and recording of all the activities of Honeypot. It deceives the intruder by capturing all activity within honeypot without attacker knowing about any monitoring i.e. with introduction of decoy systems.

2.5 Monitoring Methods of Low-Interaction Honeypots

2.5.1 Mwcollect (Malware Collection Tool)

Mwcollect (Swanson, 2008) is a low-interaction honeypot. This honeypot is installed on top of the operating system. The Mwcollect daemon is responsible to open well-known ports often used for purpose of attacking

by malware. Simulation of vulnerabilities of the open ports lures attackers who would then exploit these ports and send their malware shell codes to the Mwcollect daemon. The daemon is responsible for interpreting the shell code, parsing the exploited packets and take necessary action to download the malware which is then added to the repository for further analysis. Additional, shell codes can be written to extend the functionality offered by Mwcollect.

2.5.2 Honeyd

Honeyd (Provos, 2003) has been developed by the University of Michigan and is a daemon that can used to simulate a large network on a single network host. It is a framework for creating virtual honeypots using unused IP addresses of a network, which simulates various operating systems and services. Honeyd uses arpd tool to route all illegitimate traffic to an unused IP address and presumes the every connection made to this unused IP address is a candidate for an attack. The virtual hosts communicate with the attacker. According to (Provos, 2003), Honeyd is simulated at stack level, hence tool such as nmap cannot get fingerprint of the honeypot server. The creation of virtual hosts in a configuration file allows analyst to open TCP and UDP ports, bind scripts to those ports (if required) and bind IP address to a port. The facility to create customised scripts and binding them to ports to handle connections is very useful functionality for virtual hosts.

2.6 Monitoring Methods of High-Interaction Honeypots

2.6.1 Sebek

Sebek is a high interaction honeypot system that works as follows for the purpose of monitoring:

- Sebek installs as a loadable hidden kernel module that would capture all host activities. As a result of installation, Sebek, replaces a number of sensitive system calls in the original operating system. For instance, in the latest Sebek development for Linux 11 system calls have been replaced viz: sys_open, sys_read, sys_readv, sys_pread64, sys_write, sys_writev, sys_pwrite64, sys_fork, sys_vfork, sys_clone, sys_socketcall (Jiang and Wang, 2007). The hashtable for system calls is updated / hijacked by Sebek with its own system handlers as shown in Figure 2.3.

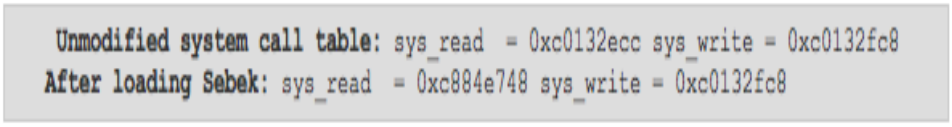

Figure 2.3: Instance of modified sys_read system call after loading of Sebek

- Upon successful replacement of system calls by Sebek, it would intercept any subsequent invocations of above mentioned system calls and capture the arguments as well as any context information such as PID. After capturing, Sebek invokes system call handlers and execute the system call together with passed arguments in order to complete requested service call.
- All collected information about invoked replaced system calls would be sent to remote Sebek server so that it can analysed in real time or saved for later analysis.

Figure 2.4, shows the Sebek based approach to honeypot monitoring. For the purpose of monitoring the malicious activity in the honeypot, the internal sensors like Sebek need to be transparent and tamper-resistant. However, as mentioned before, it case of comprise, attacker may introduce anomalies such as (Jiang and Wang, 2007):

- modification of replaced system call table,

- inconsistency in statistics transmitted by honeypot,
- Unsebek (Corey, 2003) of a honeypot system.

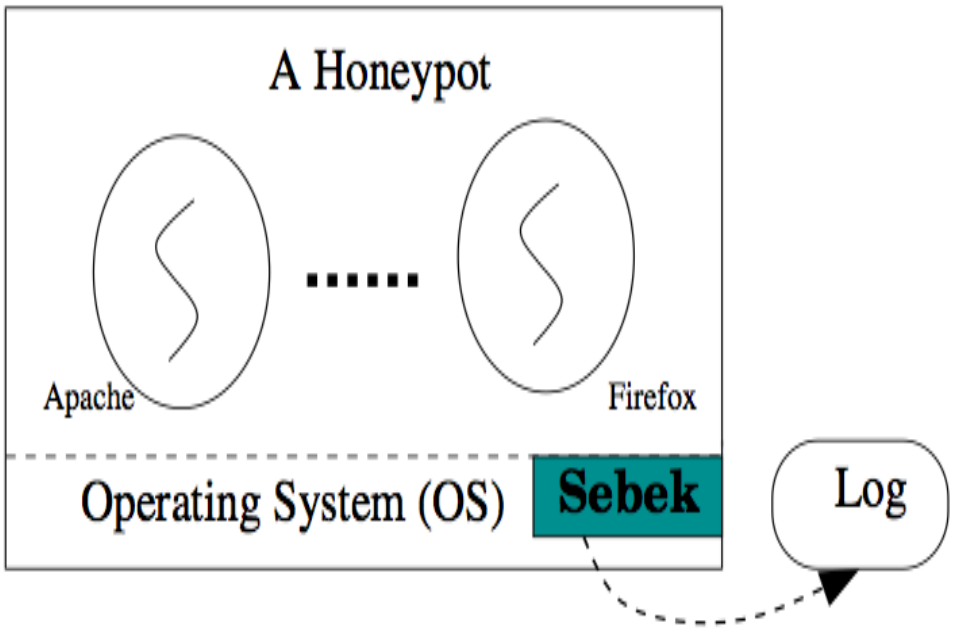

Figure 2.4: Sebek based approach in honeypot monitoring in context of HTTP (Source: (Jiang and Wang, 2007))

2.6.2 Honeynets

Honeynet is a high interaction honeypot developed by The Honeynet Project (Project, 2012) in order to capture information on the network. The primary purpose of the honeynet is to gather information on security issues. It acts as a gateway called Honeywall, by collecting data from and to the honeypots on the network.

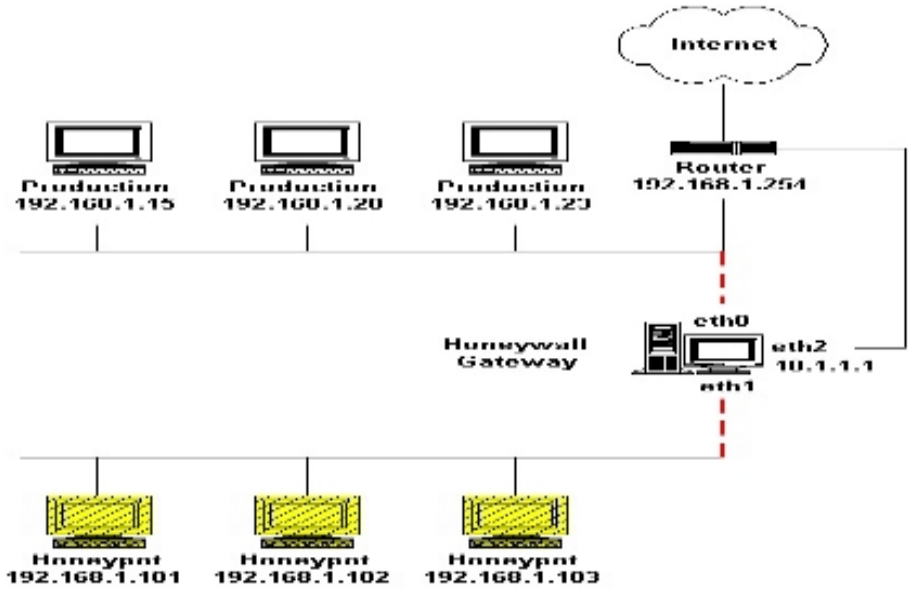

Figure 2.5: Honeywall architecture (Source: (Project, 2012))

Figure 2.5, shows the honeywall gateway that forms the main part of the Honeynet and work by capturing all the traffic entering or leaving the honeypot network. It separates the honeypots victims from rest of the network. According to (Meijerink and Spellen, 2006), it can be configured as layer 2 or layer 3 routing gateway, however layer 2 configuration is preferred as in bridge mode it is difficult to be detected by the attackers as the gateway would not have any IP address associated with itself. A highly controlled network where every packet entering or leaving is monitored, captured, and analysed consists of data control, data capture and data collection.

- Data Control: In a scenario where a honeypot deployed within honeynet is compromised, honeynet have to contain all the activities and ensure that production systems are not harmed in anyway. It should be ensured that all traffic can flow in and out of honeynet without attackers detecting control activities (Meijerink and Spellen, 2006).
- Data Capture: This part captures all activities within the honeynet and the data entering and leaving the honeynet without attacker knowing that they are being monitored. All the activities of the attacker are logged

and the captured data is analysed to understand vulnerabilities and motives of the attacker.

- Data Collection: All captured data is forwarded to a centralised data collection point. This facilitates captured data to be collected, analysed and archived at one location.

2.6.3 Argos

Argos (Portokalidis et al., 2006)is a high interaction honeypot that is based on Qemu (Bellard, 2005). Qemu is a fast machine emulator for various CPUs including x86, PowerPC, ARM and Sparc and on hosts including x86, PowerPC, ARM, Sparc, Alpha and MIPS (Bellard, 2005). Argos is known for fingerprinting zero-day attacks for instance worms and other malware without a requirement for any user input (Portokalidis et al., 2006).

As seen in Figure 2.6, all the incoming network traffic is logged into the network trace database and concurrently sent to an unmodified application running on top of the operating that is based on Qemu fast emulator. The emulator uses dynamic taint analysis to check a vulnerability that is being exploited in order to change control flow of the application. This is achieved by tagging all the data originating from unsafe source as tainted, track this tainted data during execution and prevent usage of tainted data in addition to its identification. All locations where the tainted data is copied i.e. memory or registers are also tagged as tainted locations. Argos can raise an alarm whenever instructions such as call, ret, jmp, longjmp etc. are invoked. Upon alarm, Argos starts by dumping all tainted blocks as well as information indicating all addresses that triggered violations into a log file. This step is known as signature creation process. In addition, extra information such as executable name, open files, used sockets, network ports etc. are also gathered as part of forensics. Argos creates the signatures based on the collected inputs and sequence of bytes known as flow signature that could be submitted to IDS. In addition to creation of signatures, Argos

has Sweetbait – a system that correlates collected signatures that have been collected at various sites to create the final signature for a malware.

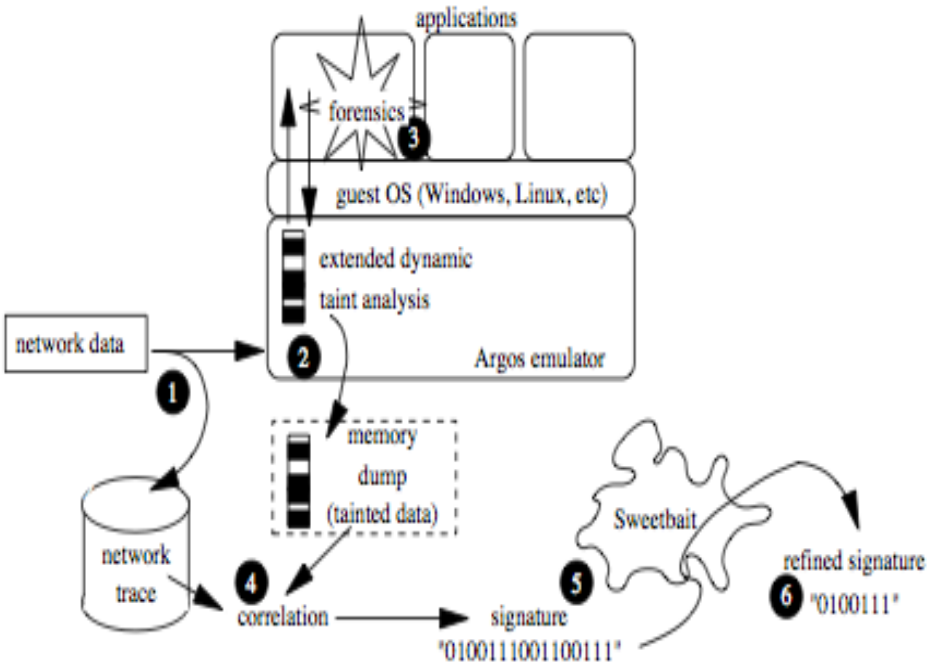

Figure 2.6: High-Level overview of Argos (Source: (Portokalidis et al., 2006))

2.7 Advantages and Disadvantages of Honeypots

Upon understanding about background and detection of honeypots, following distinct advantages have been realised as compared to other security systems (Project, 2012):

- Small Data Sets: Honeypots are always interested in the traffic that arrived to them rather than the traffic overload that is generally observed in production systems, where it is difficult and complex task to differentiate between legitimate and illegitimate packets. Overall, it collects small data sets of high value.
- Catch new attacks, false negatives: As honeypots capture everything arriving to them, they are capable of catching new tactics and attack methods which may previously be considered false negatives.

- Work in encrypted or IPv6 environments: Honeypots have been tested to work with encrypted traffic as well as have scaled to IPv6 environments.
- Minimal Resources: As only limited data is captured, a high-end set of resources is not required in case of honeypots. It is a simple concept that requires minimal resources.

Some of the disadvantages associated with honeypots as compared to other security system / approaches are as follows (Project, 2012):

- Limited field of View (Microscope): It is inherent to honeypots that the only activity or data captured by them is when the attacker directly interacts with them. Attacks happening on the other parts of honeypot network is unknown to a particular honeypot.
- Risk (mainly high-interaction honeypots): Though unlikely in low-interaction honeypots but in case high interaction honeypots, as the deployment of a real operating systems and applications is committed, in scenarios of compromise, parts of production network may be attacked that could be a major concern for an organisation.

2.8 Intrusion Detection Systems

2.8.1 Overview

Intrusion is described as an attack or attempt to sidestep security mechanisms of computer or networks, or compromising the confidentiality, integrity, or availability (CIA) (Bace and Mell, 2001). Weber (1998) categorized attacks into 5 classes: Denial-of-Service (DoS), Probing attacks, User to Root (U2R) attacks, Remote to Local (R2L) attacks, and Data attacks. In the first class, computing resources are overwhelmed by attackers in order to handle legitimate users' requests (Labib, Vemuri, 2008). Probing is described by Paliwal and Gupta (2012) saying:

"an attack in which the hacker scans a machine or a networking device in order to determine weaknesses or vulnerabilities that may later be exploited so as to compromise the system"

U2R attack is defined as an attempt by hackers exploiting weaknesses in the system in order to obtain root user privileges. R2L attack is an attempt from a remote machine to get unauthorized local access. In the last class, malicious code is injected in data looking normal that passes firewalls to attack and destroy systems.

According to SANS Institute (2001), the intrusion detection process is involved in observing and analysing user and system activity, reviewing both system configurations and vulnerabilities, evaluating the stability of critical system and data file, reporting abnormal activities, and carrying out system audit. This process is carried out by a software application or hardware system. There are three components for intrusion detection systems; Network Intrusion Detection system, Network Node Intrusion detection system, and Host Intrusion Detection System. The first component (NIDS) is in charge of scanning traffic from and to all machines over the network (Bradley, 2014). The second component (NNIDS) is responsible for examining and analysing traffic directed from the network to a specific host (SANS, 2001). The last component (HIDS) checks incoming and outgoing packets from a host only and notifies users or administrator of suspicious activities (Bradley, 2014).

There is a common mistake that intrusion detection and prevention systems are considered as alternatives for a firewall. Although they are used in the context of network security, they are totally different. Both are used in conjunction to improve a network security. Hassan (2011) explained this difference in a very simple form saying:

"We can think a firewall as security personnel at the gate and an IDS device is a security camera after the gate"

A firewall is used to prevent intrusion between networks by restricting the access between them but it is not used to report or find attacks or threats inside networks. On the other hand, IDS is responsible for finding and reporting unwanted entries to the system.

2.8.2 Detection methodologies

Intrusion detection methodologies are categorized into three types; signature based detection (SD), Anomaly-based detection (AD), and stateful protocol analysis (SPA). The first methodology (SD) defines a pattern that matches a particular attack. This methodology is very effective to find known attacks or threats. However, it is not easy to keep patterns up to date. In addition to this, this methodology is not effective in detecting unknown threats or attacks (Liao et al. 2012). On the other hand, the second type (AD) is very effective in finding new vulnerabilities. AD works on the basis of defining the network behaviour (profile). Then, the defined profile is compared with monitored events and activities to detect significant attacks. The main disadvantage of this methodology is its high dependency on profile definition, not well-defined profiles can lead to weak accuracy in detecting attacks or threats (Jyothsna et al. 2011). The third category (SPA) works similarly to AD but with generic profiles defined by vendors. Those preloaded profiles are related to specific protocols. Therefore, the system will be able to find unexpected sequences of commands like issuing a command repeatedly (Scarfone and Mell, 2007). However, it will not be able to cease attacks behaving as benign protocol (Liao et al. 2012). The table 2.2 summarizes advantages and disadvantages of each methodology.

By looking at pros and cons of the IDS methodologies mentioned above, we will find that providing more effective detection for attacks or threats can be achieved by using hybrid methodologies. For example, using SD and AD together will provide a system that can detect both known unknown attacks (Liao et al. 2012)

	Signature based detection	Anomaly-based detection	Stateful protocol analysis
Advantages	Efficient in finding known attacks or threats	Effective in discovering new vulnerabilities	The system will be preloaded with generic profiles created by vendors.
Disadvantages	Difficulties in keeping patterns up to date. Not effective in detecting new vulnerabilities	Its effectiveness is highly dependent on profile definition.	Cannot detect attacks behaving as benign protocol

Table 2.2: Advantages and disadvantages of IDS methodologies (Liao et al. 2012)

2.8.3 Limitations of Intrusion Detection Systems

IDS play an important role in finding possible attacks or threats and have a significant positive impact in security infrastructure. However, it is not an answer to all issues related to security as there are some limitations. One of those limitations is inability to trace and analyze all traffic on highly loaded or busy networks (SANS Institute, 2001). Therefore, the system may not be able to provide an instantaneous report for attacks or threats in such scenarios. It is also reported on the same paper (SANS, 2001) that IDS does not help if there is weakness in a network protocol, or in the absence of strong identification and authentication mechanism.

Another limitation addressed by Rebecca and Mell (2001) is lacking the capability of conducting investigation in the absence of human interaction. They also mentioned that it is not effective in dealing with

switched networks. A study conducted by Excamilla and Terry in 1998 reported a number of issues with IDS. One of those issues is that some IDS do not provide verification for the checksum field in the IP header. This shortcoming gives hackers a chance to manipulate this field. As a result, the system will record different information than what it should receive. Moreover, it was found that IDSs are not cheap solutions as it consumes different types of resources during both setup and monitoring phases. In addition, it demands high level of technical and organizational expertise. In spite of the requirements of a lot of resources and expertise, it is not simple to trace the improvement in security processes (Werlinger et al. 2008). A common complaint reported is that IDS can generate enormous number of alerts while the majority of those alerts are false positive (Ho, et al. 2012).

2.8.4 IDS Tools

SNORT tool is a widely used source network intrusion prevention and detection system built by Source fire. It is an example of signature based IDS. Numerous researchers have evaluated the performance of this tool. One example of such a study carried out by Rani and Singh (2014) concluded that SNORT managed to find 12 signatures one of them is ICMP PING attack having the max numbers of alerts reported by SNORT. Another study (Salah et al. 2011) carried out to evaluate the performance of SNORT when using Windows 7 and Windows 2008 server, reported the following:

"Setting the scheduling priority to favour either kernel processing or user applications has little or no impact on SNORT's performance under both normal and malicious traffic."

Although they obtained good results with SNORT, they reported that its performance is affected by heavy network traffic and this is one of IDS limitations mentioned above. Although SNORT is very popular over other products, easy to deploy, has a wide community support, and can run on most operating systems, other tools have some advantages over SNORT

that need to be considering when selecting an IDS/IPS. One of those tools is Suricata which can run multi threads while SNORT can run a single thread, supporting multi-thread gives Suricata the advantage of using more than one CPU. Kachal and Shevade (2012) said about this advantage:

"Suricata has the advantage that it can grow to accommodate increased network traffic without requiring multiple instances. SNORT is lightweight and fast but limited in its ability to scale beyond 200-300 Mbps network bandwidth per instance"

Despite of having this advantage over SNORT, the multi-thread architecture consumes more memory and CPU usage as reported by Albin (2011) in his comparative analysis of SNORT and Suricata.

BRO is another IDS tool that is worth to be looked at. Mehra (2012) enumerated some advantages of BRO over SNORT in her brief study and comparison of SNORT and BRO. One of those advantages she mentioned is the ability of Bro to operate effectively on high-speed networks while SNORT does not work perfectly on high-speed networks. In addition to this advantage, it was reported in that study that the Bro signatures are more sophisticated than the signatures used in SNORT. On the other hand, it was found that BRO is difficult to deploy compared to SNORT. Moreover, BRO can run only in UNIX environment while SNORT can run in most popular operating systems.

2.8.5 Evaluation Metrics of IDSs

According to (Lazarevic et al., 2003, Zanero and Savaresi, 2004), anomaly detection rate and false alarm rate are the best measures that can be used for evaluation of IDSs. Clearly, the detection rate is equivalent to efficiency and the false alarm rate refers to effectiveness of IDSs. The anomaly detection rate is the ratio of number of detected intrusions to the total number of intrusions that were introduced into the network traffic as show in the equation below (Ertoz et al., 2004).

$$Efficiency = \frac{True_{positives}}{All_{alarms}}$$

Where

True Positive: Malicious traffic correctly classified by IDS.

False Negative: Malicious traffic incorrectly classified by IDS.

All Alarms: True positives plus false negatives.

It is clear from the equation that as the value of efficiency approaches 1 (more capable of detecting illegitimate traffic), IDS becomes more efficient. While, false alarm rate refers to the false-positive rate of IDSs i.e. number of legitimate network traffic that have been analysed by IDS as intrusions as shown in the equation below (Ertoz et al., 2004).

$$Effectiveness = \frac{True_{positives}}{All_{positives}}$$

Where

True Positive: Malicious traffic correctly classified by IDS.

False Positive: Normal traffic incorrectly classified as malicious.

All Positives: True positives plus false positives.

This effectively refers to all anomalies that have not been detected by the IDS. In addition to, efficiency and effectiveness, (Sommers et al. 2004), further added two metrics viz. central processing unit (CPU) utilisation and packet loss. These measures are useful with regard to evaluation of IDS in terms of handling traffic load.

2.8.6 Offline Evaluation

According to (Lippmann et al., 2000, Mahoney and Chan, 2003), another method of evaluating IDSs is the method where datasets of network traffic that includes attacks is used for evaluation without requirement to create the network topology. It is the use of tools such as TCPDUMP that

are used for such evaluation. The common datasets available for the purpose of evaluation of IDSs include the data set created by Defense Advanced Research Projects Agency (DARPA) in 1998 and 1999 known as 1998 DARPA set and 1999 DARPA set.

DARPA sets are simulations of network traffic that include attacks thus offering blind evaluation material for researchers (Mahoney and Chan, 2003). According to (Lippmann et al., 2000), these data sets have been captured at border of network on external router interface, hence are not filtered or subjected to any intrusion detection techniques.

Figure 2.7 shows the conceptual view of DARPA evaluation test bed. The 1998 DARPA set includes 7 weeks of labelled data and 2 weeks of unlabelled data where approximately 300 instances of 38 different attacks exist. The 1999 DARPA has approximately 5 million connections over 5 weeks out of which 3 weeks include attack vectors. Table 2.3 shows categories of various attacks within DARPA set (Lippmann et al., 2000).

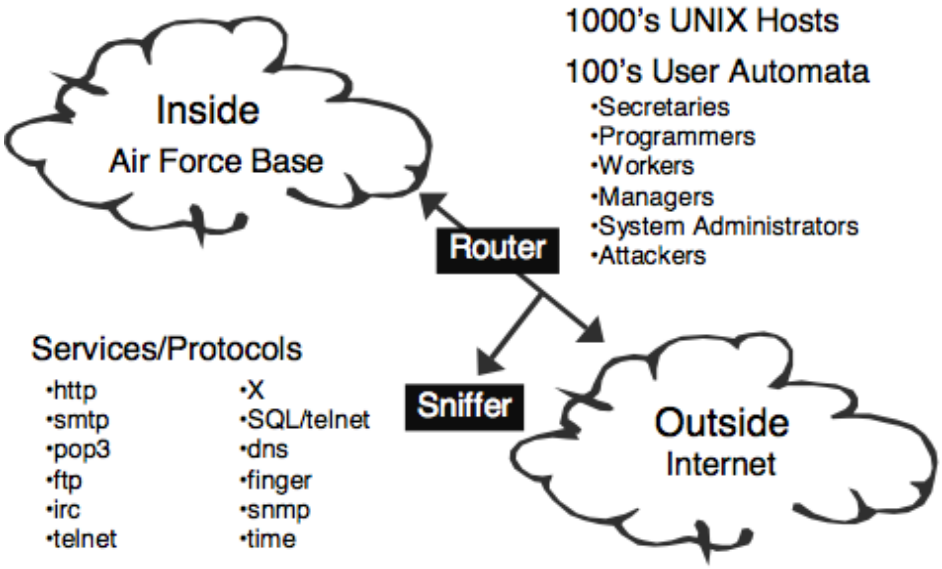

Figure 2.7: Conceptual view of DARPA evaluation test bed that create 1000's of virtual hosts and 100's of users to simulate a small Air Force base separated by router from the Internet (Source: (Lippmann et al., 2000))

The advantage of DARPA sets is that it enables fast trial runs for evaluation of IDS. Furthermore, these trial runs are also identical as the data set used for evaluation of IDSs is same. From researcher's perspective, it saves them important resources and time as they are not required to collect data sets as these DARPA sets are free to use. However, some of the shortcomings of using these data sets have been highlighted in (Nikolova and Jecheva, 2011, McHugh, 2000, Mell et al., 2003) as follows:

1. Network topology used for collection of these data sets is too simple.
2. The target systems are small in number.
3. Low background traffic.

	Solaris	SunOS	Linux	Cisco Router
Denial of Service	Apache2 Back Mailbomb Neptune Ping of death Process table Smurf Syslogd Udp-storm	Apache2 Back Land Mailbomb Neptune Ping of death Process table Smurf Udp-storm	Apache2 Back Mailbomb Neptune Ping of death Process table Smurf Teardrop Udp-storm	
Remote to Local	Dictionary ftp-write guest http-tunnel phf xlock xsnoop	Dictionary ftp-write guest phf xlock xsnoop	Dictionary ftp-write guest imap named phf sendmail xlock xsnoop	Snmp-get
User to Root	At Eject Ffbconfig Fdformat Ps	Loadmodule	Perl Xterm	
Surveillance / Probing	Ip sweep Mscan Nmap Saint Satan	Ipsweep Mscan Nmap Saint satan	Ipsweep Mscan Nmap Saint Satan	Ipsweep Mscan Nmap Saint Satan

Table 2.3: Attack types in evaluation data set (Source: (Lippmann et al. 2000))

2.8.7 Online Evaluation

Upon considering the shortcomings presented by (McHugh, 2000) in context of offline evaluation of IDSs, it has become important to generate data sets that include realistic network traffic as well as attack vectors. Another tool form Lincoln Labs called as Lincoln Adaptable Real-time Information Assurance Testbed (LARIAT) is capable of generating realistic background traffic as well as generate real network attacks (Rossey et al., 2002, Allen, 2007). It has been created in order to answer the shortcomings of DARPA sets, where objective has been to support real-time evaluation and a test bed that is configurable as well as easily deployable. LARIAT simulates both internal and external network traffic making it possible for IDS evaluation in both network environments. Another two tools namely Harpoon and Malicious Traffic Composition Environment (MACE) developed by (Sommers et al., 2004) are similar to LARIAT. Harpoon is used for generating flow-level network traffic based on real network packet traces while MACE is performance benchmarking and malicious traffic generating tool (Sommers et al., 2005, Sommers et al., 2006). Table 2.4 shows taxonomy of MACE exploits (Sommers et al., 2005).. According to (Lo et al., 2010, Sommers et al., 2006), a new release of tool that combines Harpoon, MACE and best features of DARPA set called Trident has number of additional features that are useful for evaluation of IDSs. Table 2.5 shows the list of Trident tools for NIDS performance evaluation (Sommers et al. 2006).

Host Based					Network Based
Application Level		Transport Level			
Worms	Backdoors	DoS	Fragmentation	Other DoS	
Welchia Nimda CodeRed2 Blaster Dameware Sasser	Mydoom Sdbot	winnuke	Rose Teardrop1 Teardrop2 Bonk Nestea Oshare	Synflood Pod Land jolt	Smurf Fraggle

Table 2.4: Taxonomy of MACE exploits (Source: (Sommers et al., 2005))

Name	Description
Attack-replay	A flow replay tool that allows two-way replay of a packet trace.
Autom-gen	A script that stores service descriptions and generates service-specific automata for Harpoon.
Exec-grom	A traffic grooming algorithm that uses trust heuristics to separate benign traffic from suspicious traffic.
Payload-gen	A tool that reads a groomed packet trace and outputs packet pools that corresponds to automata states.
Payload-sanitize	A tool that sanitizes inconsistencies in protocol headers that are introduced due to interleaving.
Split-darpa	A script to separate malicious DARPA traffic from benign based on labels.
Harpoon plugin	A traffic generation plugin for Harpoon that executes the service description automata to produce application payload traffic.

Table 2.5: Trident tools developed for NIDS performance evaluation (Source: (Sommers et al. 2006))

(Gadelrab et al., 2007) and (Saber et al., 2011) have presented framework for defining test scenarios. (Saber et al., 2011) has argued in the paper that current classification of attacks do not cover of requirements of evaluating IDSs. They have provided framework for covering all characteristics of attacks in order to have complete evaluation.

2.9 Data Mining

2.9.1 Overview

Data mining is defined as the process aiming to find useful information from large data sets (Tan, Steinbach, Kumar, 2006). Grossman (1997) defined data mining in a more descriptive definition saying:

"Concerned with uncovering patterns, associations, changes, anomalies, and statistically significant structures and events in data"

Based on those definitions, we can consider data mining as an analytical tool that helps users to look at data from different angles in order to categorize them.

Data mining is used in many different applications. One of those applications is intrusion detection. Reddy and Rajulu (2011) reported that data mining can have a great contribution in helping IDS to focus on malicious activities and real attack by removing normal activities from alarm data. They also added that it can play an important role in identifying bad sensor signatures or false alarm generators

Gabra, Baha-Eldin, and Korshi (2014) reported that they managed to reduce number of irrelevant alerts by 99.9% when they used of one of data mining based method for classifications. Another research that aimed to enhance IDS alarm quality by using a new data mining technique concluded that using this technique reduced the alarm load by 82% (Al-Mamory, Zhang, 2008). The idea of this technique is to produce clusters and categorize alarms, then each cluster abstracted as a generalized alarm. The generalized alarms linked to root causes are transformed to filters in order to decrease alarms load in the future. Different data mining techniques will be discussed in more detail in next sections.

2.9.2 Data mining limitations

Despite of the promising results mentioned above, Phung (2000) reported that building an effective solution using data mining faces some obstacles. One of those obstacles is the massive increase in the amount and complexity of data to be analysed, this issue makes data mining quite expensive in terms of computation. Data mining in this case may consume a lot of CPU and memory resources that are expensive or not available. Phung also added that carrying out analysis on network traffic using a sample of the data and not all of them for the purpose of generating profiles may cause false conclusions.

2.9.3 Genetic algorithms

Genetic algorithm is considered as machine learning method based on the principles of evolutionary computation (Reddy, 2011). Genetic algorithm has been used in many different applications with promising result. In the context of intrusion detection, Kumar and Guyal (2004) said:

"They incorporate the concept of Darwin's theory and natural selection to generate a set of rules that can be applied on a testing set to classify intrusions".

Mujahid and Mathew (2014) discussed the advantages of genetic algorithm in their research about this technique. One of those advantages is its capability to find a solution for any optimization problem. In addition, they reported that it is capable of handling multiple solution search spaces.

Many researchers applied genetic algorithms in intrusion detection research area with very high success rate. One of the researches in the area of network anomaly detection used both multi-agent and genetic algorithm (Crosbie, Spafford, 1995). Another research that combines two techniques; genetic algorithm and fuzzy data mining was conducted by Bridges and Vaughn (2000). A hybrid algorithm developed by Castro and Zubin (2002) to achieve the optimization of intrusion detection by using both of support

vector machines and genetic algorithm. Genetic algorithms were used by Goyal and Kumar to identify and classify different types of attack connections, they succeeded in lowering false positive rate to 0.2%. Another genetic algorithm model, that achieved a low false rat, was developed by Chittur (2001).

Although genetic algorithm achieved promising results in the research area of intrusion detection, Majeed and Kumar (2014) addressed some limitations in their survey about genetic algorithm in intrusion detection systems. One of those limitations is the complexity to propose a problem space. They also added that selecting the optimal parameters for genetic algorithm is not a simple process. In addition, it was mentioned in this survey that systems based on genetic algorithms are not easy to configure. Moreover, that survey reported that it is required to have a local searching technique in conjunction with genetic algorithm for effective functioning.

2.9.4 Artificial Neural Network

It is a computational model based on the principles of animal's central nervous systems. This model has the capabilities of machine learning as well as pattern recognition. Anthony (2014) described it as a system that adapts its structure in the learning phase; this adoption is based on external or internal information flowing through the system. There are two types of neural network algorithms; supervised and unsupervised training algorithms. The first algorithm learns expected output for a specific input. The most common used architecture of type is Multi-Level Perceptron (MLP). This architecture is widely used in solving pattern recognition problems. On the other hand, unsupervised training algorithms can learn without the need to specify expected output. One of most popular unsupervised algorithms exploited in solving classification problems is Self-Organizing Maps (SOM).

The concept of neural network, adaptive learning, attracted many researchers to work in the area of using neural network in intrusion detection. Some tests using neural network were made by Lippmann and Cunningham (2000) at MIT Lincoln Laboratory. Multi-Level Perceptron (one of neural network structures) is used to find host attacks, and attacks that try to get root-privilege on a server by looking at specific keywords linked to attacks on network traffic. By applying that approach, they managed to reduce false alarms by two orders of magnitude. They also reported that they managed to raise the detection rate to around 80 % with the DARPA data base. This approach is able to catch old as well as new attacks not contained in the training data sets. Another research that exploited MLP was conducted Ghosh and Schwartzbard (2000). They reported that when they had applied MLP to anomaly detection, they obtain good result, 77% of attacks were detected and 3% of alarms were categorized as false alarms. On the other side, they obtained high false rate when they applied MLP on misuse detection. Girardin employed SOM to carry out clustering of network traffic and detect attacks. His approach managed to catch IP spoofing, FTP password guessing, and network scanning. Kukiełka and Kotulski (2009) concluded in their research about adaptation of the neural network-based IDS to new attacks detection that neural networks in their experiments succeeded to classify the network traffic similar to the traffic presented in the learning stage. On the other hand, it did not manage with a good accuracy to classify new attacks and new normal traffic that are different than the traffic existed during the training phase.

2.9.5 Naive Bayes

This technique is considered as a simple probalistic classifier based on Bayesian probability model. Its simplicity comes from that it estimates the class probabilities by assuming that features or attributes are

conditionally independent (Tan et al. 2005). Panda and Parta (2007) described naive bayes classifier saying:

"The naïve Bayes classifier operates on a strong independence assumption. This means that the probability of one attribute does not affect the probability of the other"

Amor et al. (2004) mentioned in their research (Naive Bayes vs. Decision Trees in intrusion detection systems) that the main advantage of this technique is its simple structure. This simplicity helps in constructing the mode incrementally. As a result, it will be easy to be updated. On the other side, its performance is very poor with some datasets that have a strong dependency between features due to the strong independence relation assumption that is not always true in the real world (Ji, Yu, Zhang, 2011).

Panda and Patra (2007) carried out some tests in network intrusion detection exploiting naive bayes classifier. They reported that testing the system they developed using 10% KDDCup'99 data set achieved 95% detection rate. In addition, the model was built in very short time (1.98 sec). However, they noticed that it generates false positives with a higher rate compared to propagation neural network. Another experiment aiming to improve intrusion detection by employing naive bayes was run by Taruna and Hiranwal (2013). They reported that the system they proposed managed to increase the balance detection rates for 4 attack classes; DoS, U2R, R2L, and probe. They also reported that the system also generates false positive at acceptable level. Sagane and Dhande obtained similar results when they followed a similar approach (2014).

2.9.6 Decision Tree

Decision tree Classifier is known as a simple and popular technique employed in solving classification problems. It is defined as (Prediction Works, 2011):

"A predictive modelling technique from the fields of machine learning and statistics that builds a simple tree-like structure to model the underlying pattern of data"

Markey (2011) enumerated advantages of decision tree over other classification techniques, one of the main advantages is that it generates a set of rules which are transparent, easy to understand, and easily employed into real-time technologies as Intrusion Detection systems. However, Rokach and Mimon (2014) pointed out that this technique works only with target attributes having discrete values. They also said:

"The greedy characteristic of decision trees leads to another disadvantage that should be pointed out. This is its over–sensitivity to the training set, to irrelevant attributes and to noise".

Bouzida and Cuppens (2008) compared the results obtained when testing intrusion detection based on decision tree with the results obtained when using neural network. They concluded that employing decision tree in intrusion detection are more effective in detecting new attacks compared to neural network. Jain and Upendra (2012) proposed a model based on an enhanced version of C4.5 decision tree in order to detect attacks. They tested the proposed model on 10% of KDD data set and found that it catches attack with 96.9% accuracy. Another experiment employing C4.5 decision tree algorithm carried out by Bidgoli, Analoui, Rezvani, and Shahhoseini (2008) showed that the proposed system managed to detect probe attack with 100% accuracy, it was also able to detect DOS attacks with accuracy tending to 100%. On the other hand, the system detects U2R and R2L with low accuracy. Abbes, Bouhoula, and Rusinowitch (2004) used protocol analysis approach based on decision tree to solve the false negative issue occurring in pattern matching processes. Kailashiya and Jain (2012) developed a model based on decision tree in conjunction with stratified weighted sampling. They tested their proposed system using KDD cup dataset and found that they obtained a good accuracy rate at 93.85% and

error rate at 3.92%. Makkithaya, Reddy, and Acharya (2008) proposed a fragmentation based c-fuzzy decision tree model. Their goal in their research was improving the performance by selecting more suitable data set and decreasing the number features. They reported that the results they obtained proved that the model could be used to build an effective intrusion detection system.

2.9.7 K Means

K Means is a clustering technique that partitions data objects into K clusters dependant on their feature values. Tan et al. (2006) explained K Means saying:

"k-means defines a prototype in terms of a centroid, which is usually the mean of a group of points, and is typically applied to objects in a continuous n-dimensional space".

This clustering technique is very simple to understand and to be employed in implementing solutions that solve clustering problems (Vora, Oza, 2013). However, Derban and Moldovan (2006) reported some disadvantages that may represent obstacles in obtaining optimal solutions to when using K Means clustering technique. One of those disadvantages is that the algorithm is not capable of specifying the number of clusters (K), this means that there is a need to set this number of by users. This also means that users need to carryout experiments with different number of clusters to obtain the best results. Another shortcoming mentioned by Derban and Moldovan is that there is a high dependency on initial centroids in partitioning data objects.

Singh (2010) conducted a research on intrusion detection using K Mean algorithm. The approach proposed in that research was tested with 1998 DARPA audit data, the best result was obtained when setting number of clusters (K) to 2 with a detection rate tends to 96% and low false positive rate. Wei at el. (2011) developed an enhanced version of K Means

algorithm that solves some issues in K Means algorithm. They described this algorism saying:

"In the improved k-means algorithm clustering guiding function is introduced. It can help the algorithm determine clustering in direction of the high point density"

It was found that testing the developed model with KDD 99 increased the detection rate by 2.94%, it also reduced the false positive rate by .76% compared to the K Means algorithm.

Table 2.6 summarizes advantages and disadvantages of data mining techniques mentioned in this chapter.

Technique	Advantages	Disadvantages
Genetic Algorithm	- Finding a solution for any optimization problem. - Handling multiple solution search spaces.	- Complexity to propose a problem space. - Complexity to select the optimal parameters - The need to have local searching technique for effective functioning
Artificial Neural Network	- Adapts its structure during training without the need to program it.	- Not accurate results with test data as with training data
Naive Bayes Classifier	- Very simple structure. - Easy to update.	- Not effective when there are high dependency between features.
Decision tree	- Easy to understand - Easy to implement	- Works effectively only with attributes having discrete values. - Very sensitive to training sets, irrelevant features and noise.
K Mean	- Very Easy to understand. - Very simple to implement in solving clustering problems.	- Number of clusters is not automatically calculated. - High dependency on initial centroids.

Table 2.6: Advantages and disadvantages of data mining techniques

2.9.8 Related Research Works to the first contribution

One of the most well-known works in context of using data mining for intrusion detection is by Axelsson (1999), the model created uses the maximised posterior probabilities as parameters, provided by Bayesian algorithm. As a result the false alarm rate that is usually shown by IDSs is

reduced. In our research the aim has been to create new signatures based on reasoning of outlier instances. In other words, our research is complementing research work by provision of a reasoning module.

Another research study conducted by Abraham (2001) uses real-time network intrusion detection systems for detection of misuse. It employs association rules; characteristic rules and Meta rules to provide results, with regard to deviation from normal network activity.

Lee and Stoflo (2000) outlined a data-mining framework for constructing intrusion detection models. The key idea is to first apply data mining programs, to audit data to compute frequent patterns, extract features, and then use classification algorithms to compute detection models.

Chang (2007) used the method of back propagation by sample query and attribute for intrusion detection, to identify and analyse features of training data. The main contribution of that research paper has been a reduction in processing time and storage of data instances.

Perhaps the closest research that has been conducted is by Barbara (2003), where a training system was built to classify unknown and false alarm instances. Furthermore, Barbara (2003) used and analysed the unknown instance by following its audit trail, in order to provide a concrete result, informing if the data instance was outlier. Our research is going a step further by creating a rule signature whereby an IDS rules holder can be updated automatically.

2.9.9 Related Research Works to the Second Contribution

A number of researches conducted in the multi stage attacks detection area. One of those researches was conducted by Alserhani et al. (2010). The proposed correlation framework in that research combines two engines; online and offline, and two mechanisms; high quality knowledge-based and statistical-based correlation. The online tools receive alerts from IDS then it recognizes multistage attacks using defined rules provided by the offline

engine. The proposed framework achieved 92% multi stage detection rate and 21.8% false positive rate during their lab experiments for 35 multi stage attack scenarios. This approach reduces the computation expenses by analysing only alerts received by IDS. However, this massive dependence on alerts received by IDS may lead to missing capturing attacks in case of not receiving alerts. Another research that follows event correlation approach was carried out by Spadaro (2011). In that research, false alerts were reduced by combining both signature and anomaly based IDS to remove redundant events. In addition, some classifiers were trained with different attack categories to carry out an early classification for the logged security events. Moreover, meta data is combined with event data to reduce false positive rate.

Other efforts made by Templeton (2010), they proposed a system that follows Attack scenario construction approach. This approach is based on associating two security incidents, it tries to find consequences of one incident and prerequisites of the incident that may occur later. The strength point of this approach is the ability to construct new attacks created by a mixture of known attacks can be detected. On the other hand, attacks cannot be tracked without finding causes and effects of attacks. Moreover, it requires large consumption of computer resources.

Another research was carried out by Ourston et al. (2012). The research is based on using Hidden Markov Models (HMM). The idea of using HMM is to determine the most likely attack type corresponding to a sequence of alerts received by IDS. This study found that HMM approach achieved greater classification accuracy compared to other approached. However, they reported that the obtained accuracy was associated with the expense of additional computations.

The proposed methodology in this thesis has an advantage over those solutions by not being dependant on receiving alerts from IDS the mentioned solutions above. It also does not require a complex computation

and memory resources compared to them. In addition, the mentioned solution requires an update with sequences of new attacks while the proposed system focused on the source of the attack not the attack logic. However, this may represent an issue if an attack comes from an IP address not classified yet as suspicious. Moreover, the throughput of the proposed methodology is relatively low compared to other solutions due to using web services that consumes sometime to get IP information.

2.9.10 Weka data mining tool

It is a widely used software tool in machine learning written in Java and built at the University of Waikato, New Zealand. It includes a large number of machine learning and data mining algorithms. This tool has become very popular in the academic and industrial fields. This can be run on different platforms as it is written in Java programming language. In addition, it is free as it is under General Public License (GNU). Moreover, it has a graphical user interface which makes it easy to use. Despite of those advantages, Weka cannot handle datasets larger than a few megabytes, it issues an out of memory error (Naudts, 2004).

Many researches and studies were conducted to evaluate the performance of different algorithms using Weka data mining tools. One of those studies, which was carried out by Wahbeh et al. (2013), gives a comparative analysis between data Mining Tools over some classification methods. They reported the following:

"According to study the functionality built into to Weka and available through add-ons makes the software highly robust for a variety of users"

2.10 Feature selection

Feature selection is defined as the process of obtaining a subset of related attributes or features to be used in constructing a model. In other

words, it removes inappropriate, irrelevant or redundant data; this behavior can play an important role in improving learning accurateness, and recovering result unambiguousness (Kamepalli and Mothukuri, 2014). One of widely used feature reduction algorithms is correlation-based feature selection where subsets of features are assessed on the basis of the hypothesis stated by Hall (1999):

"Good feature subsets contain features highly correlated with the classification, yet uncorrelated to each other"

Hall also claimed the following in his research about CFS for machine learning:

"Feature selection for classification tasks in machine learning can be accomplished on the basis of correlation between features, and that such a feature selection procedure can be beneficial to common machine learning algorithms" There are also other widely used feature selection reduction algorithms such as information gain (IG), gain ratio (GR) and Fast Correlation Based Filter (FCBF). Although feature selection techniques have a positive impact as mentioned earlier, Batal (2014) pointed out that some features that are looked at as irrelevant may be useful when associating them with others.

Chou (2007) proposed a CFS algorithm to select a subset of most relevant features. The result was retrieving six data sets from UCI databases and an intrusion detection benchmark data set, and DARPA KDD99. Those data sets are then used to train and test C4.5 and naive bayes algorithms. They reported that the proposed approach achieved the highest averaged accuracies compared to CFS and FCBF. Chae and Choi (2014) developed a feature selection method based on attribute ratio that uses attribute average of total and each class data. The results of experiments conducted in that research showed that there is that between accuracy and attribute ratio value. They reported that the highest accuracy (99.794%) was achieved when 22 features were used.

2.11 Discussion

So far, several papers have been published to address the security threats (Meijerink and Spellen, 2006, Singh and Joshi, 2011). But none of these solutions, neither antivirus nor firewalls, can totally prevent these attacks. Therefore, the design of an Intrusion Detection System (IDS) that is facilitated by honeypots that is expected to detect and mitigate threats and attacks on production networks has become a priority for researchers. Such system would not only allow production networks to protect themselves from security threats but would also autonomously create evidence for forensics analysis in case of attack.

The honeypots analysed have the capability to record and monitor network activity (legitimate or illegitimate). Though the logs are mere collection of the network activity and require forensic / network data analyst to analyse the logged data. Honeyd and Mwcollect (low-interaction honeypots) have ability using configuration files to emulate vulnerabilities associated with certain open ports. Honeynet and Argos do not emulate vulnerabilities and are real operating systems with real services where methods include data control, data capture and data collection in case of Honeywall and signature creation of tainted malware based on various inputs.

Literature for honeypots and intrusion detection systems as isolated subject has been focus of many research works. However, researchers have not invested effort into facilitation of IDS using honeypots to secure production networks. Much progress has been made within IDS for purpose of detecting known attack vectors. However, little has been made in the area specifically related to use of high interaction systems for autonomously updating IDSs. Honeypots or monitored honeynet networks can be used for the purpose of (Mokube and Adams, 2007):

1. Defensive distraction system in order to direct an attacker towards machines containing no valuable information;

2. An early warning system that can inform about exploitation trends to IDS; and
3. A data collection store that can be used to examine the methods and processes of exploitation of a honeypot in order to create forensic reports, when required.

Issues with IDS in terms of quality include metrics of effectiveness, adaptability and extensibility (Nazer and Selvakumar, 2012). An IDS can be effective, if it has high intrusion detection i.e. large rate of true positives (ability to realise that the certain network activity is actually an attack) as well as if it has low rate of false positives (ability to realise that a given network activity is not an attack and considered normal network activity). Generally, this is achieved by creation of rules by the expert based on domain knowledge and / or analysis of logged network data making it a very complex process (Kayacik et al., 2012). An IDS can be adaptable, if it can detect variations in previously known exploits and update the rules seamlessly in order to prevent intrusion. The literature has indicated various intrusion detection methods but none has been found to be adaptable where unknown attacks that are "child of" known attacks can be realised autonomously (Amro et al., 2012). An IDS can be extensible, if it allows integration of additional modules or updating / customisation of existing modules by the administrator. According to (Nazer and Selvakumar, 2012), customisation in current IDS is difficult as expert rules and statistical measures as environment specific.

It has been realised that in addition to issues listed above, the gap between collection of network data and creation of rule / signature is not only large in terms of temporal terms (i.e. the time it takes to create a new signature that would be informed to IDS) but also in terms of automated intelligent data analysis tools that could upon analysis create signatures with high true positive and low false positive rates. Honeypots are not solutions for intrusion detection and hence they do not have production value. The only value that is offered by them is that they help with data collection and

provide initial set of data so that intelligent rules can be created that can prevent intrusions on network or organisational assets. The topic honeypot has been discussed in this chapter, as it was possible solution of data collection. The objective was to use actual live data for the purpose of research study however, due to filtering (possibly offered by an Internet Service Provider (ISP)), the amount of attacks on vulnerable resource was minimal and the most that was received on resource was DNS queries that were unimportant. Hence, the methodology was altered to use KDD data as opposed to data collected from honeypot.

2.12 Summary

Currently, the research is focused on study of various high interaction honeypot tools as well as capabilities of data capturing tools. The focus is now shifting towards creation of an autonomous data analysis tool that would be based on data mining techniques and would take input of raw data collected as a result of hybrid of host-based and network-based monitoring tools. In this chapter, the researcher has provided a concise overview on honeypots and their uses. The chapter also discussed various classifications and categories of the honeypots namely research, production, low-interaction and high interaction honeypots. A detailed description of detection methods used in high interaction honeypot systems viz. Sebek, Honeynet and Argos as well as preparation of low-interaction honeypot system using configuration files has been discussed. Although, honeypots have been active area of research for a decade, but they are gaining popularity due to degree of analysis tools and capturing and detection techniques that are becoming invaluable in the world of cybercrime and network forensics.

This chapter has also discussed IDS including the difference between a firewall and IDS, intrusion detection methodologies, and intrusion detection limitations. It also gives a quick look over IDS tools

highlighting advantages and disadvantages of each tool. In addition, this chapter has gone through data mining, some techniques used in data mining, and how those techniques employed in intrusion detection by different researchers to improve the performance of IDS. This part of the thesis has also provided an overview about feature selection and how it can have a great contribution in helping learning machine algorithms exploited to improve the performance of IDS.

CHAPTER 3

MULTI-LAYER CLASSIFICATION SYSTEM

3.1 Introduction

In the context of information technology, intrusion can be defined as a series of attempts in order to compromise security of a network-based resource (Liao, 2013). Network-based systems or resources require constant monitoring in order to ensure that malicious activity can be contained (Kang, Fuller, Hanover, 2005). An Intrusion Detection System (IDS) is responsible for monitoring network traffic and based on a set of rules raising alerts for information security officers, when malicious traffic is detected (Kang, Fuller, Hanover, 2005).

Although IDSs are successful in terms of preventing attacks on network resources, they are not adaptable in cases where new attacks are made, i.e. they need human intervention for investigating new attacks (Borah, Chakraborty, 2011) (Roesh, 1999). Furthermore, an IDS could become a bottleneck, where it is employed on a busy network. IDS requires time for

processing network data, before it can be released to a production network (Roesh, 1999).

One possible solution for addressing the above problem is to create a system that is based on Machine Learning. This signature-based system will use existing IDS, such as SNORT, for comparing packet signatures to rules defined by SNORT, and the packets found to be malicious are subjected to being passed to an intelligent model that has been trained to detect malicious content (Roesh, 1999) (Kim, Lee, Kim, 2014). Hence, SNORT will act as a first level of filter reducing the amount of traffic for further investigation using the intelligent model. Overall, this reduces the load on SNORT, hence providing a reduction in analysis at the SNORT level, and further reducing human intervention, as the intelligent trained model is responsible for deciding if a certain set of packets are malicious or otherwise. If a set of packets are found to be malicious, an automated signature will be created that will update the set of rules used by SNORT.

The novelty is offered by integration of the training model for detecting misuse in the incoming network data packets with a reasoning model that is applied on outliers (uncategorised data packets) (Kim, Lee, Kim, 2014). The result of the reasoning model is in the form of a rule that can then be used by an IDS, on a production level system, to filter automatically malicious data packets of the type just identified.

The aim of this chapter is to provide a comparative study of classification of algorithms for the purpose of creating a training model used for misuse detection. In this comparative study we present results in terms of a confusion matrix and metrics such as true-positive, false-positive, true-negatives and false-negatives. Also presented are comparisons between expected and predicted classes of KDD'99 (KDD'99, 1999) intrusion detection data by a random split of 66% for creation of the training model, and 34% for testing of the training model for misuse detection.

Section 2 gives an overview of the proposed methodology and Section 3 provides a description of the KDD'99 intrusion detection data set, discussing metrics used for attribute selection in the KDD'99 intrusion detection data set. Section 4 discusses the classifier module, providing a brief background with regard to classifier algorithms, namely Naïve Bayes and Decision Tree, then goes through the experiments results. Section 5 discusses the reasoning module and the experiments that have been conducted, and then discusses the results obtained. Section 6 provides a conclusion to this chapter.

3.2 Classification Approach

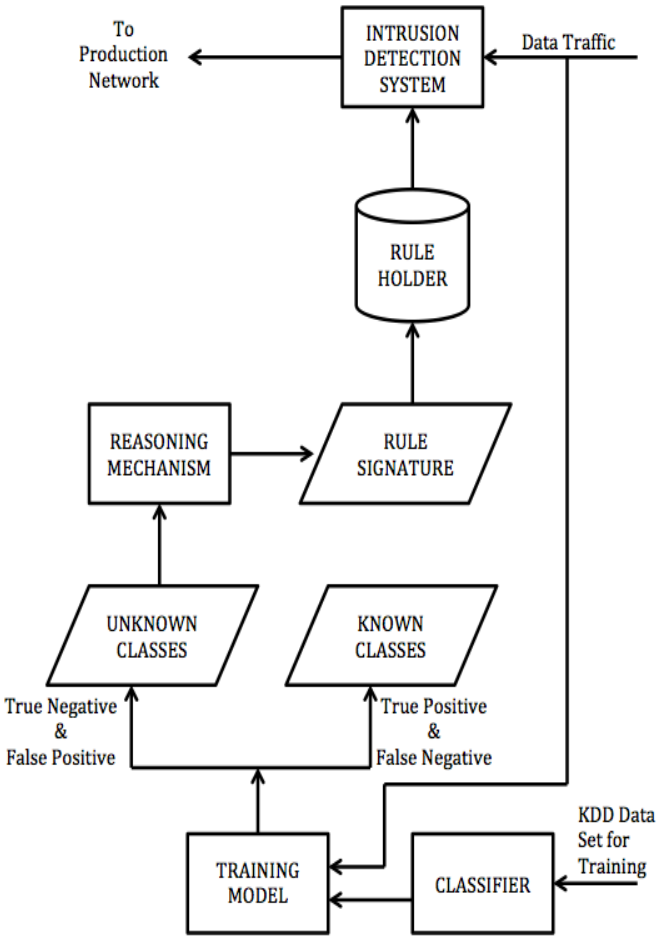

Figure 3.1: High Level View of Research Process

The research process consists of the following elements (see Figure 3.1):

- Intrusion Detection System: SNORT will be used in this solution as a signature based IDS. In addition to its main functionality as an IDS, it will be used as a network sniffing tool that feeds the training model with the live traffic.
- Rule Holder: This contains all signatures used by SNORT to capture attacks matching the stored signatures.
- Data Set and Categorisation: The first step in the research process is to find a reliable high quality network traffic data set, where each packet has been labelled so that the training model is created, and as a result the classification can be used reliably.
- Feature Selection: The network packets in the data set are then passed through an attribute evaluator, in order to extract a set of features that can be used effectively to detect intrusions. Non-essential features are known to be not only a bottleneck in terms of cost of computation, but are also factors that contribute towards increased error rates (Wei, Wang, 2011).
- Classifier Module: This module is responsible for building a classifier using Decision Tree and Naïve Bayes that can compute a model using the most discriminating features in an instance of a data packet, in order to describe a class (concept). This is done by training a classifier, using a pruned set of features, where the objective is that the model created is more generic than the rules (as compared to SNORT) and hence, it outperforms this in accuracy and effectiveness, when compared to general rule-based signature matching systems.
- Training Model: This model is the outcome of the classifier module. The results of each classification algorithm will be compared to each other then one of them will be selected to be used as the training

model. This model will classify the traffic to either known or unknown classes. The traffic will be passed to the reasoning module in case if it is unknown for further investigation.

- Reasoning Mechanism: The purpose of this mechanism is to provide another stage for classifying the network traffic, if the first stage fails to classify it. The reasoning mechanism is based on a hybrid model built using neural network (MLP) and fuzzy logic. The outcome of this module will be in a form of a signature that will be added to the rule base.

3.3 Data Set – KDD'99

3.3.1 Overview

The data set used in this chapter is the KDD'99 intrusion detection data set. This data set is based on a 1998 DARPA initiative and has been used by researchers for evaluation of various intrusion detection methodologies in the past. The data are collected as a result of a setup of a fictitious military network with a number of target machines running various services. A sniffer has been used to record all network data using raw TCP/IP dumps (KDD'99, 1999).

The data set consists of 41 discrete and continuous attributes and has 22 attack classes and 1 normal class, where each instance in the data set has been categorised as one class.

The attacks are further divided into 4 categories:

- Denial of Service Attack Category (DoS) where a target host is compromised by the request of service from a multitude of machines. (e.g. syn flood)
- User to Root Attack Category (U2R) where an attacker attempts to get unauthorised access to root level of a target system. (e.g. buffer overflow attacks)

- Remote to User Attack Category (R2L) where a hacker tries to take control of a remote machine by exploiting vulnerabilities of the system. (e.g. guessing password)
- Probing Attack Category (probe) where an attacker scans the machines (generally on a network) in order to collect useful information (for instance, services running) about those machines. (e.g. port scanning)

1.3.2 Features of the Data Set

For selection of important attributes in the network data set, Correlation-based Feature Selection (CFS) was employed. A search algorithm, as well as a classifier function, is used by CFS to evaluate the importance of each feature and provide a subset of features (Hall, 1999). The heuristic that is used by CFS describes important features that are highly correlated to the class; however they are uncorrelated from each other (Hall, 2009). In a data-mining context, this approach is based on information gain that measures the importance of each attribute for predicting class, based on the calculated entropy of that attribute. An attribute with entropy value approaching 0 will have information gain approaching 1 (Davis, Clark, 2011).

$$H[D] = -\sum_{j=1}^{|C|} P(c_j) \log_2 P(c_j) \qquad (1)$$

Where C is the desired class

Information gain by removal of an attribute can be computed as a difference of entropy before removal to entropy after removal of that attribute (Davis, Clark, 2011).

$$gain(D, A_i) = H[D] - H_{A_i}[D] \qquad (2)$$

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

Search Method:
        Best first.
        Start set: no attributes
        Search direction: forward
        Stale search after 5 node expansions
        Total number of subsets evaluated: 514
        Merit of best subset found:    0

Attribute Subset Evaluator (supervised, Class (nominal): 42 class):
        Classifier Subset Evaluator
        Learning scheme: weka.classifiers.trees.J48
        Scheme options: -C 0.25 -M 2
        Hold out/test set: Training data
        Accuracy estimation: classification error

Selected attributes: 1,3,5,8,14,23,35,36,37,38 : 10
                     duration
                     service
                     src_bytes
                     wrong_fragment
                     root_shell
                     count
                     dst_host_diff_srv_rate
                     dst_host_same_src_port_rate
                     dst_host_srv_diff_host_rate
                     dst_host_serror_rate
```

Figure 3.2: Attributes selected upon using CFS evaluator and depth first search

3.4 Classifier Module

3.4.1 Overview

For misuse detection, we have used two classifiers, namely Naïve Bayes and Decision Tree, for creation of a training model. This section provides a brief description of each technique. For a more detailed background, the reader can study Bhargava (2013) or Rawat and Jain, (2013).

3.4.2 Naïve Bayes

Bayesian reasoning is applied to decision-making that deals with probabilistic inference, i.e. the knowledge about previous events is used to

predict future events (Altwaijry, 2013). In a Naïve Bayes classifier the availability or unavailability of a certain attribute is not related to availability or unavailability of another attribute. Naïve Bayes provides an advantage when making decisions based on small amounts of training data, to compute mean and variance of the attributes in order to compute its class. Bayes theorem provides a method of calculating the posterior probability, P(c|x), from P(c), P(x), and P(x|c). A Naive Bayes classifier assumes that the effect of the value of a predictor (x) on a given class (c) is independent of the values of other predictors (Rawat, Jain, 2013). This offers conditional independence. Bayes algorithm is explained by the following:

$$P(c \mid x) = \frac{P(x \mid c)P(c)}{P(x)} \quad (3)$$

$$P(c \mid X) = P(x_1 \mid c) * P(x_2 \mid c) * ... P(x_n \mid c) * P(c) \quad (4)$$

- $P(c \mid x)$ is the posterior probability of class given attribute.
- $P(c)$ is the prior probability of class.
- $P(x \mid c)$ is the likelihood which is the probability of attribute given class.
- $P(x)$ is the prior probability of attribute.

3.4.3 Decision Tree

A decision tree classifies a data set through a sequence of decisions, where a decision on a current node facilitates a decision to be made on the following nodes (Bhargava, 2013). This forms an n-array tree structure, where a decision is made by traversing from a root node to a leaf node, where the leaf node represents a class. Nodes are formed of attributes or features from the data set. C 4.5 is one of the most common algorithms to create a decision tree. Ruggieri (2002) described how C4.5 constructs a tree as follow:

"The algorithm constructs a decision tree starting from a training set T S, which is a set of cases, or tuples in the database terminology. Each case specifies value for a collection of attributes and for a class. Each attribute may have either discrete or continuous values. Moreover, the special value unknown is allowed, to denote unspecified values. The class may have only discrete values."

J48 (Bhargava, 2013) – is an open source implementation of the C4.5 algorithm for decision trees, available through Weka (Hall, 2009). J48 offers handling of a variety of input data types, for instance nominal, textual and numeric, and is high in performance. The algorithm operates as follows:

- *The algorithm operates over set of instances used for training, C.*
- *If all instances in C are in class P*
    - *Then create a node P and end.*
    - *Else select attribute F and create division node.*
- *Partition the instance C into subset of values ($V_{1..n}$) for attribute F.*
- *Apply the algorithm recursively to each of the subsets of instance C.*

3.4.4 Experiment Environment

For this chapter, 10% of the whole KDD'99 intrusion detection data set was used for training; this small subset was selected randomly and represents the complete KDD data set. This data set represents a concise version of the whole data. This data set contained approximately half a million classified instances of network data packets (KDD'99, 1999). For the purpose of testing the effectiveness of thes model created, 34% of this data set, which approximates to 168,000 of known classified instances, was used. A training model has been created after considering two model creation strategies:

1. All-Classes: In this case, a training model has been created by considering all the classes described in the KDD intrusion detection data set.

2. Two-Classes: In this case, a training model has been created by categorising the KDD data set into only two different classes, namely normal and malicious.

3.4.5 All-Classes Based Model Creation Strategy

Table 3.1 shows the results of instances, classified correctly and incorrectly, upon use of Naïve Bayes and Decision Tree (J48) algorithms, for creation of a training model using an all-classes model creation strategy. It is noted that results of correctly classified instances, by employing Decision Tree, are slightly better than Naïve Bayes (Bhargava, 2013) (Altwaijry, 2013).

Instances Classified	Naïve Bayes	Decision Tree
Correctly	91.82% (154228)	99.95% (167890)
Incorrectly	8.18% (13739)	0.04% (77)

Table 3.1: Results for Naïve Bayes and Decision Tree Using an All-Classes Model Creation Strategy

Figure 3.3 and Figure 3.4 show the comparison between predicted and expected classes using Naïve Bayes and Decision Tree classifiers respectively. It is clear from the graphs that in Decision Tree predicted classes are the same as the expected classes, which is observed using a 45 degree gradient. While in the case of the Naïve Bayes approach, a large amount of variance is caused by conflict between predicted and expected classes. It is further observed that Normal class packets have a large amount of jitter that is a result of incorrect prediction, using the Naïve Bayes training model.

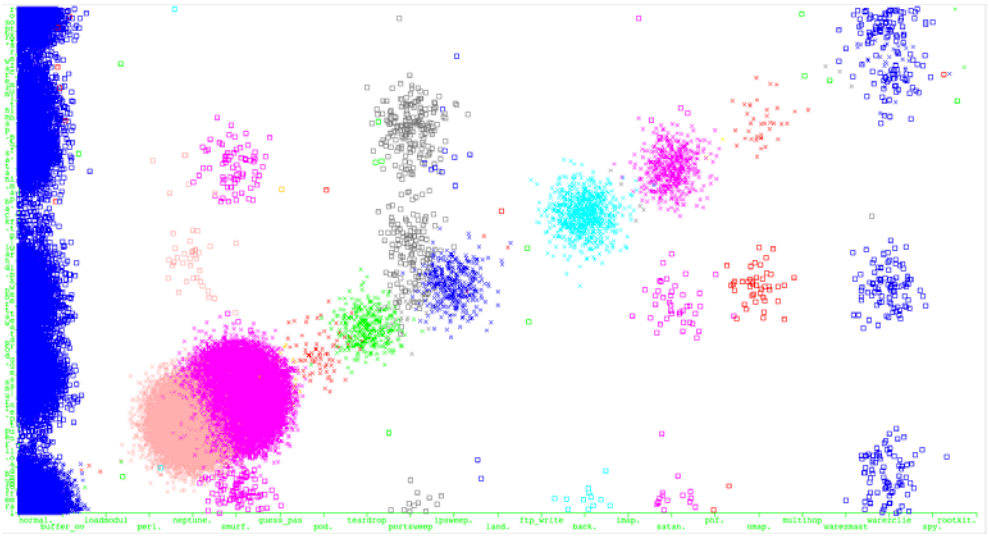

Figure 3.3: Variance of predicted vs. expected classes using the Naïve Bayes all-classes model creation strategy

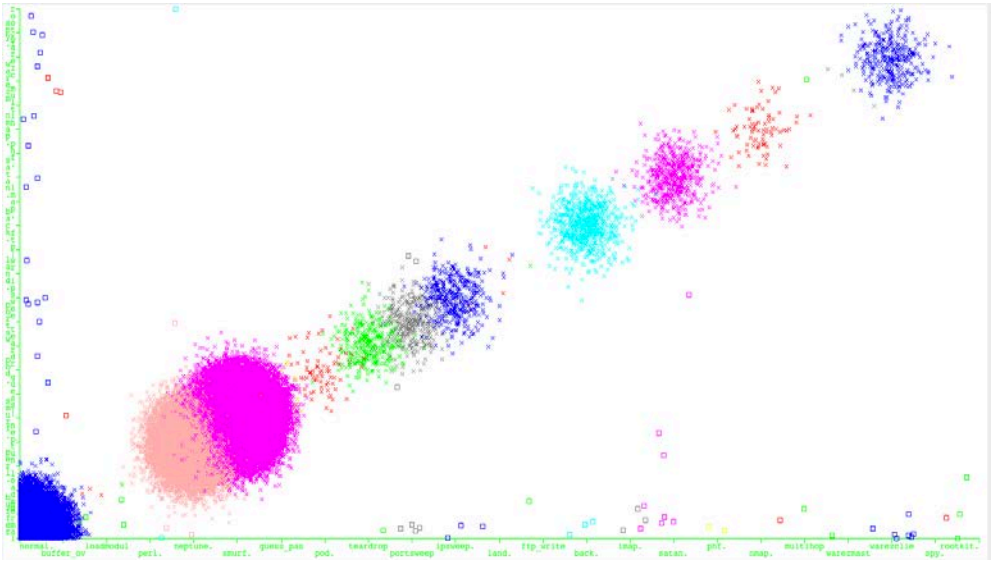

Figure 3.4: Variance of predicted vs. expected classes using Decision Tree all- classes model creation strategy

Table 3.2 shows the consolidated comparative results per class for each classifier for metrics True-Positive (TP) and False-Positive (FP). It is observed that, although Naïve Bayes has a high TP, it is skewed by results of

FP. On the other hand, the Decision Tree generates almost a high consistent TP and a consistently low FP.

Class	True-Positive		False-Positive	
	Naïve Bayes	Decision Tree	Naïve Bayes	Decision Tree
Normal	0.617	0.999	0	0
Buffer Overflow	0.462	0.615	0.001	0
Load Module	0.4	0.2	0.001	0
Perl	0	0	0	0
Neptune	0.999	1	0.001	0
Smurf	0.998	1	0	0
Guess Passwd	0.952	1	0.025	0
Pod	0.987	1	0	0
Teardrop	0.988	0.997	0	0
Portsweep	0.111	0.979	0.01	0
IPsweep	0.97	0.993	0.007	0
Land	0.75	1	0	0
FTP Write	0	0.5	0.002	0
Back	0.984	0.996	0	0
IMAP	1	0.4	0	0
Satan	0.894	0.986	0.002	0
PHF	1	0	0.011	0
NMap	0.457	0.988	0.001	0
Multihop	0	0	0.006	0
Warezmaster	0.75	1	0.002	0
Warezclient	0.107	0.979	0	0
Spy	0	0	0	0
Rootkit	0.667	0	0.012	0

Table 3.2: Accuracy / Class for Naïve Bayes And Decision Tree Using All-Classes Model Creation Strategy

The results also show that a high FP rate has been observed in classes corresponding to probing, and in remote to local attack categories. This

indicates the attacks were mainly in categories where the attacker is scanning the machine to understand more about vulnerabilities of network resources, and furthermore that these could be initiated from remote locations, where a local machine could have been compromised.

3.4.6 Two-Classes Based Model Creation Strategy

Table 3.3 shows the results of instances classified correctly and incorrectly upon use of the Naïve Bayes and Decision Tree (J48) algorithms for creation of a training model using a two-classes model creation strategy. It is observed that using training models created by use of Decision Tree is better than Naïve Bayes.

It is further observed that in contrast to the all-classes-based model creation strategy, Naïve Bayes has performed better in correctly classifying the instances in the two-classes-based model creation strategy, as the results are improved from 91.82% to 98.44%.

Instances Classified	Naïve Bayes	Decision Tree
Correctly	98.44% (165349)	99.96% (167898)
Incorrectly	1.56% (2618)	0.04% (69)

Table 3.3: Results for Naïve Bayes And Decision Tree Using Two-Classes Model Creation Strategy

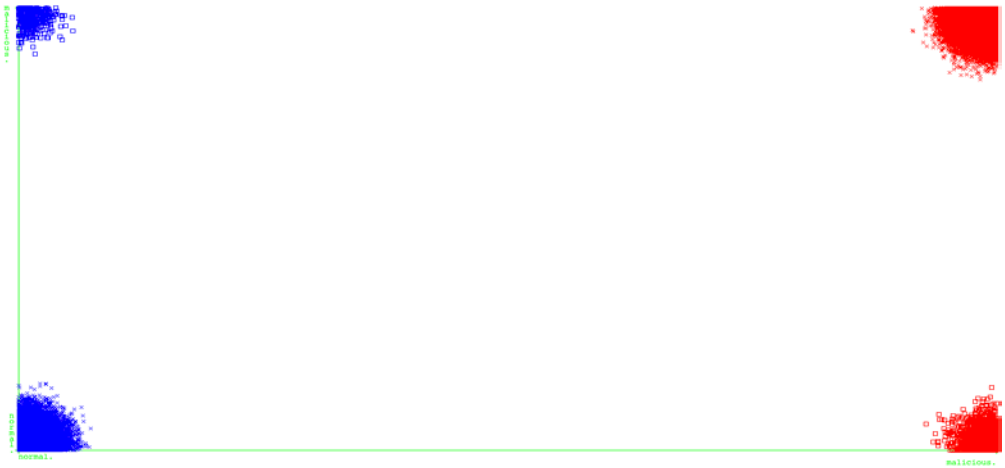

Figure 3.5: Variance of predicted vs. expected classes using Naïve Bayes two-classes model creation strategy

Furthermore, it is observed in figure 3.5 that the number of incorrectly classified instances has decreased, as shown in the cluster on the top-left and bottom-right quadrant of the graph. This can be attributed to a reduction in granularity of classes, associated with data making the model for prediction of instances more accurate.

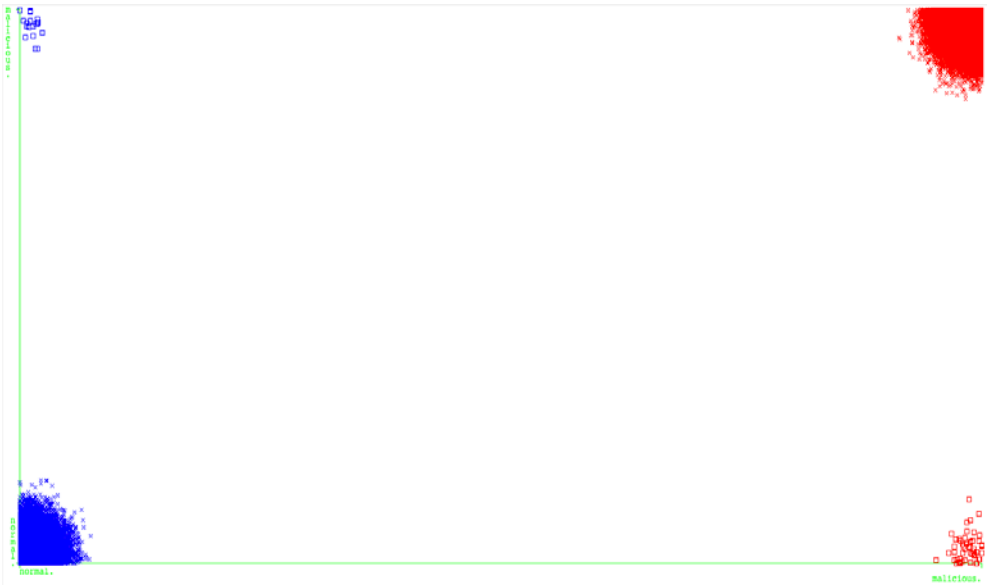

Figure 3.6: Variance of predicted vs. expected classes using Decision Tree two-classes model creation strategy

It is observed that Decision Tree has performed consistently, even with a change of model creation strategy. There has been a nominal increase in correctly classification of instances, and figure 3.6 shows the number of incorrectly classified instances decreased, as seen in the top-left and bottom-right quadrant of the graph.

Class	True-Positive		False-Positive	
	Naïve Bayes	Decision Tree	Naïve Bayes	Decision Tree
Normal	0.989	0.999	0.017	0
Malicious	0.983	1	0.011	0.001

Table 3.4: Accuracy / Class for Naïve Bayes and Decision Tree two-classes Model Creation Strategy

Table 3.4 presents the combined comparative results per class for each classifier for metrics TP and FP for the two-classes-based model creation strategy. It is observed that Decision Tree has consistently a high true-positive rate and consequently a low false-positive rate.

3.4.7 Chosen Model

Overall, it has been observed in the context of the all-classes and two-classes-based model creation strategies that the Decision Tree algorithm is more effective in prediction of classes for data instances. Furthermore, the true-positive rate of the Decision Tree algorithm is higher in the two-classes-based model creation strategy, making this two-classes-based strategy a better choice for the model. However, it should be noted that the objective of our model is to provide prediction of data instances with high granularity of categorised class, so that categorised instances can be subjected to further critique using reasoning mechanisms. Because of the aforementioned requirement, we have selected the all-classes-based training model creation strategy for prediction of classes for data instances. In other words, the all-classes-based strategy for creating a model is giving more information in context of classes without loss of correctly and incorrectly classified instances.

3.5 Reasoning Module

3.5.1 Overview

The proposed reasoning mechanism in this chapter classifies the network traffic into normal (1) or attack (0). In the other word, the mechanism is based on a hybrid model consists of two modules; the first one is based on neural network while the second one is based on fuzzy logic. Figure 3.7 gives an overview of the proposed hybrid model in this chapter. The hybrid model will classify network traffic as normal if both modules classify it as

normal while it will classify it as attack if either of the modules classifies the traffic as attack.

Neural network has the advantage of the ability to work with not complete and precise data. This merit can be employed in IDS context for detecting attacks patterns presented during the training phase but modified by an attacker in order to pass through the system (Kukiełka, Kotulski, 2010). The flexibility of fuzzy logic can be employed in case of uncertain problem of intrusion detection and allows much greater complexity for IDS (Shanmugam, 2009).

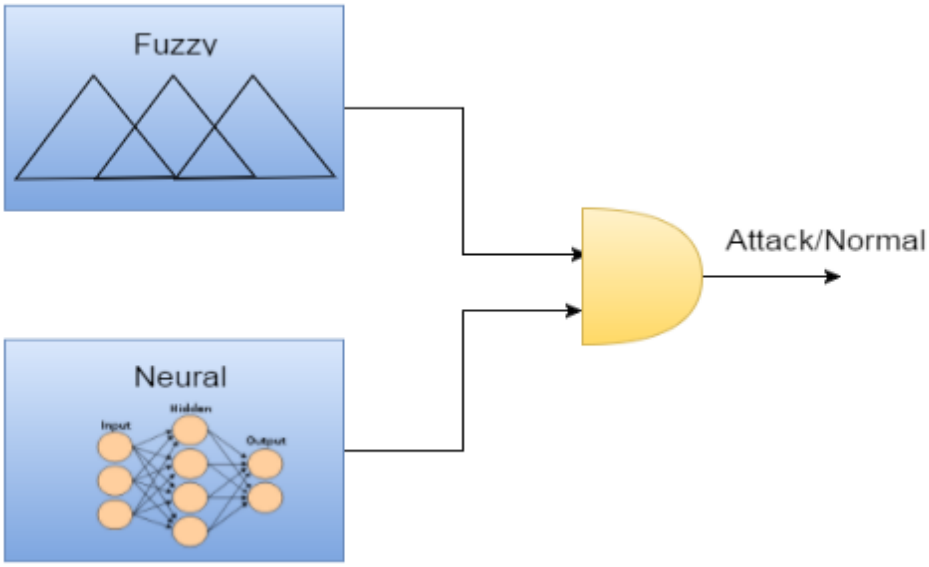

Figure 3.7: Hybrid Model Overview

The benefit of using the hybrid approach is increasing the intrusion detection rate, some of attacks may not be detected by one of the modules but the other one may be able to detect them. In other words, one module will overcome some of other module shortcomings in detecting malicious traffic. However, there is a chance of increasing the false-positive rate for malicious traffic.

3.5.2 Neural Network

As described in the previous chapter, artificial neural network is a computational model inspired by the principles of an animal's central nervous systems. This model has the capabilities of machine learning, as well as pattern recognition. It has been described as a system that adapts its structure in the learning phase; this adoption is based on external or internal information flowing through the system (Anthony, Bartlett, 2009). The most commonly used architecture of supervised neural networks is Multi-Level Perceptron (MLP). That architecture contains a number of layers (one input layer, a number of hidden layers, and one output layer), each layer contains a number of processing units called neurons. Each neuron is connected with a weight to a neuron in the following layer. MLP uses the back propagation algorithm in the training process. In that algorithm, the input data is passed to the neural network. Then, the output of the network is compared to the desired output to compute the error. That error is used to adjust the weights, in order to get closer to the desired output. The error calculation and weight changes are explained by the following (Anthony, Bartlett, 2009):

$$e_j(n) = d_j(n) - y_j(n) \qquad (5)$$

$$\varepsilon(n) = \frac{1}{2}\sum_j e_j^2\,(n) \qquad (6)$$

$$\Delta w_{ji}(n) = -\mu \frac{\partial \varepsilon(n)}{\partial v_j(n)} y_i(n) \qquad (7)$$

Where,

e_j is the calculated error for neuron j.

n is the index of training data.

d_j is the desired value.

y_j is the produced value by neuron j.

ε is the error of entire output.

w_{ji} is the weight of the connection between neuron i in a layer and neuron j in the following layer.

$\mu$is the learning rate (a value between 0.2 and 0.8).

3.5.3 Fuzzy Logic

Fuzzy logic is a computational approach based on human language rules. The fuzzy systems translate the defined rules to a mathematical equivalent (Rajasekaran, Pai, 2003). Those systems, as shown in figure 3.8, consist of fuzzifier, interference engine, rules base, and defuzzifier. Fuzzy systems work as follows (Rajasekaran, Pai, 2003):

- The fuzzifier converts the crisp inputs to fuzzy set by using specified membership functions for each input.
- Based on the defined rules, the interference engine produces a fuzzy output.
- The fuzzy output is converted to a crisp value using the membership Functions defined for defuzzification.

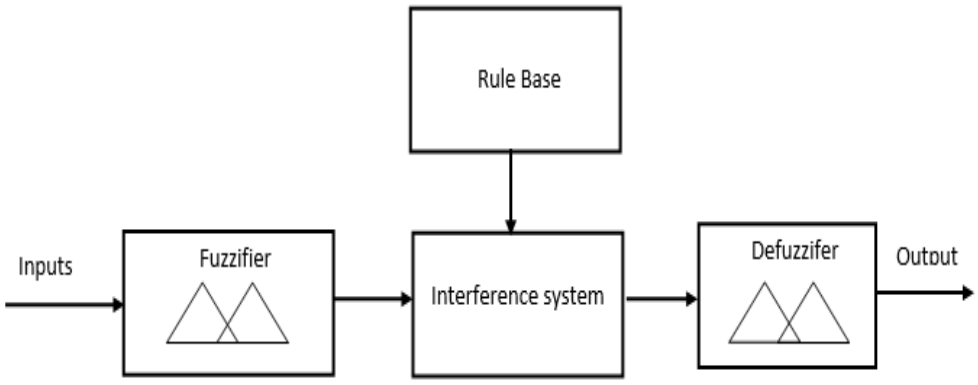

Figure 3.8: Fuzzy logic Components

3.5.4 Experiment Environment

The architecture used in the neural network module is MLP. This architecture has three layers. The first layer contains 10 neurons; the hidden

layer contains 8 neurons, with one neuron in the output layers. The neural network module was trained using 10% of the whole KDD'99 intrusion detection data set. It has been trained by setting the max mean square error to .01, and max number of epochs to 3000. All weights are initially randomly set. See Appendix B for the neural network training code. The code is based on using a neural network PHP library written by Akerboom (2007)

The training data was pre-processed as follows, before starting the training process:

1. Feature labelling: Label non-numeric attributes with numeric values. Some features are not represented by a numeric value (e.g. service), while it is required to deal with a numeric value in the neural network. For example, each service has been given a number; 1 for telnet, 2 for ftp_data.

2. Features normalisation: It has been found that each feature has a different range. Thus, all attributes have been normalised in a way that has made each attribute have the same range (between 0 and 1). This step helps in making the selected attributes comparable.

3. Remove redundant data: Removing redundant or repeated data from the training data set prevents the training algorithm to be biased in the direction of more frequent records, and ignoring less frequent records. The number of the training data set after removing the duplicates is 142,000.

The rules of the fuzzy module were created using 10%` of the whole KDD'99 intrusion detection data set as follows (See Appendix C for the fuzzy rules generation code):

1. All selected features, apart from 'service' (as it is a discrete value not continuous), have been normalised in a way that has made each attribute

have the same range (between 0 and 1). This step helps in simplifying the rules generation process.

2. We selected three values V1, V2, V3, where:

$$V1 = 0.043,\ V2 = 0.375,\ V3 = 0.75$$

3. During the iteration through the training data, each selected feature (apart from 'service') was translated from a numeric value to a description as follows:

   $0 \leq$ attribute value $< V1 \rightarrow$ Very Low (VL)

   $V1 \leq$ attribute value $< V2 \rightarrow$ Low (L)

   $V2 \leq$ attribute value $< V3 \rightarrow$ High (H)

   $V3 \leq$ attribute value $\leq 1 \rightarrow$ Very High (VH)

   The output is described as either normal or attack. The rule was then created in the following form:

   If (feature1 is feature1_desc AND feature2 is feature2_desc AND ….feature10 is feature10_desc) Then output is output_desc

4. The created rule in the previous step would not be added to the rule base, if it was previously added, to avoid having duplicate rules. The total number of rules added to the rule base is 1343. Different values had been tried for V1, V2, V3, but the value selected above gave the best results in terms of false-positive and detection rates. The last step in implementing the fuzzy module was the membership functions selection for both inputs (selected feature) and output. Figure 3.9 shows the membership functions for all inputs, apart from the 'service' feature, which was handled by a singleton membership function for each value, as it is a discrete attribute. The output has two membership functions as shown in figure 3.10.

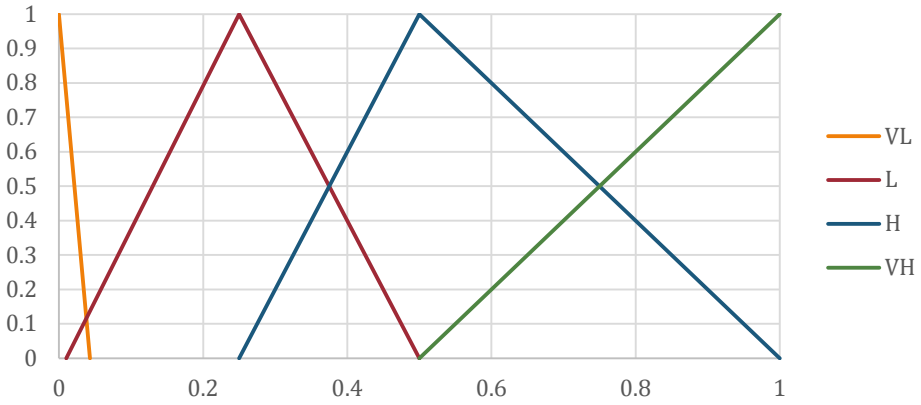

Figure 3.9: Membership function for the selected feature (not including the 'service' feature)

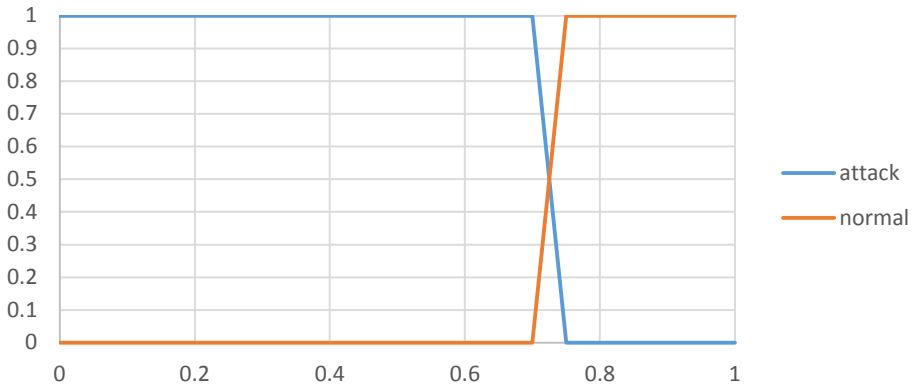

Figure 3.10: Membership function for the output

5. The results of evaluation obtained from the hybrid model, after getting the neural network training done, and completing the rules generation and membership functions selection for the fuzzy module, are shown in table 5. The evaluation process was carried out using 10% of the whole KDD'99 intrusion detection data set; this data set is different from the data set presented in the neural network training and rules generation stage.

3.5.5 Experiment Results

Table 3.5 shows that the hybrid model achieved a higher detection rate for malicious traffic than either the neural network or fuzzy logic modules, each one individually. On the other hand, the false-positive rate of the hybrid model is higher than both the neural network and fuzzy logic, each one individually. See Appendix D for the hybrid system code. The fuzzy module implementation is based on a PHP library developed by Jarzęcki (2011).

Class	True-Positive			False-Positive		
	Neural Network	Fuzzy Logic	Hybrid Model	Neural Network	Fuzzy Logic	Hybrid Model
Normal	0.971	0.978	.952	0.029	.022	.048
Malicious	0.966	.9995	.9997	0.034	0.0005	.0003

Table 3.5: Results of the hybrid model using Neural Network and fuzzy Logic

3.6 Conclusion

SNORT monitors network traffic and uses content searching and matching to detect attacks. One of the problems with using SNORT is the fact that it is not adaptable for detecting new attacks. In addition, it generates false alarms at a high rate.

Our experiments can be used to conclude that data mining can be implemented as an added portion to a pre-existing IDS. When implemented properly, data mining can improve the classification process, resulting in a

lowered number of false-positive alerts. The first stage of the proposed model built using the Decision Tree approach is able to classify most data correctly, has a better accuracy rate, detection rate and lower false-positive rate. In this chapter, we have compared two different training model creation strategies, using Naïve Bayes and Decision Tree algorithms. It is concluded that the all-classes-based training model creation using Decision Tree is the most effective, as it not only provides a better true-positive rate, but also a high level in granularity for classification of data instances, and this has high precedence over a slightly better training model created using the two-classes model creation strategy.

The second stage of the proposed model (reasoning mechanism) was built using a hybrid approach. The hybrid approach in this chapter used both neural network and fuzzy logic. The benefit of using the hybrid approach is in increasing the intrusion detection rate; some of the attacks may not be detected by one of the modules, but the other one may be able to detect them. The results obtained by that approach achieved a higher detection rate than both the neural network and fuzzy logic, each one individually. However, it has a higher false-positive rate.

CHAPTER 4

MULTI STAGE ATTACKS

4.1 Introduction

Multi-stage attacks can evolve dramatically these days, causing much loss and damage to organisations. These attacks occur through multiple steps, each step looking legal and not violating any rules. Therefore, Clark (2010) described multi-stage attacks in his research as the most challenging set of attacks to investigate and deter. He also described multi-stage attacks as follows:

"Multi-stage attacks within a single jurisdiction may permit the imposition of rules that facilitate technical solutions to attributions. We suggest that such technical solutions form a ripe area for research. But solutions to preventing the attacks of most concern, multi-stage multi-jurisdictional ones, will require not only technical methods, but legal/policy solutions as well. Better attribution techniques will neither solve nor prevent such exploitations.

Redesigning the network to accomplish robust attribution would not solve the most serious network-based cyberattacks and cyber exploitations being experienced today, which are multi-stage and multi-jurisdictional."

There is evidence of Clark's argument, when using the proposed solution in the previous chapter. The solution achieved a high level of detection rate. However, it has been found that the proposed solution works well with single-stage attacks and is not efficient when dealing with multi-stage attacks. That solution is signature-based (attributes techniques), which is not useful in the case of multi-stage attacks. It has also been found that different solutions have been introduced to stop attacks and protect organisations. However, some of those solutions ignore some communications in the network and find difficulties in differentiating between legitimate and illegal traffic, as they do not violate any rules.

Those attacks occur through multiple phases to get access to an organisation. Most of those attacks involve three phases. In the first phase, attackers try to analyse available information about the target, to find vulnerabilities and weaknesses that can be exploited. In the second phase, attackers exploit the weaknesses found in the first phase to inject malware into, or to gain access to the system. In addition, they try to get more details and conduct a deep analysis about the system to find data or resources in which they have an interest. In the final phase after gaining access, attackers destroy the system or steal valuable information (GCHQ and Cert-UK, 2015).

This chapter goes through four different multi-stage attack scenarios. The aim of this chapter is to understand the behaviour of multi-stage attacks and try to find a clue to predicting or detecting such kinds of attacks. In each scenario, the network traffic will be analysed highlighting all steps that have occurred and not been considered by

many security systems. The first scenario is about communication with a bad DNS server and how that has been employed by an attacker to register machines to its bot army. The second scenario discusses the Shady Rat attack, which is a good example showing how social engineering can be employed to target an organisation. The third scenario shows how header splitting can be employed by an attacker to target a network connected to a web host running a web application. The last scenario discussed how a vulnerable FTP service could be exploited to perform multi-stage attacks. The outcome from analysing each scenario will be in the form of rules that will be used in building a solution that will predict multi-stage attacks before they have an impact and damage organisations.

4.2 Analysis Approach

The following information will be looked at for each scenario:

IPs and URLs involved in conversations.

Operating Systems

Summary of conversations.

Based on the information extracted from the trace files and summary of conversations of traffic, the behaviour of the attacker will be modelled and some rules can be extracted to predict similar scenarios. Those rules will be used later in a proposed framework to detect such attacks.

4.3 Scenario A

4.3.1 Trace file

The communication that occurred during this scenario has been captured in a pcap file (TP Group, 2015). This file will be analysed using Wireshark to get all the information required for analysing this scenario.

4.3.2 IP Involved in the Scenario

By looking at tables 4.1 and 4.2 which contain the IP participating in the communication, it has been found that there are 268 packets and 10.129.211.13 participated in this scenario with a rate of 52.99% as a source IP address, while it participated with a rate of 47.01% as a destination IP address. In other words, 10.129.211.13 participated in all conversations during this scenario. It has been found that there are some signs that the identified IP address was compromised during this attack. One of those signs is receiving the same ICMP message in a relatively short time from different IP addresses, which indicates that there was some sort of scan occurring through it.

Source IP Addresses	Count	Percent
10.129.211.13	142	52.99%
216.234.235.165	6	2.24%
61.189.243.240	5	1.87%
10.129.102.9	4	1.49%
10.129.102.8	4	1.49%
10.129.102.7	4	1.49%
10.129.102.6	4	1.49%
10.129.102.5	4	1.49%
10.129.102.4	4	1.49%
10.129.102.3	4	1.49%
10.129.102.22	4	1.49%
10.129.102.21	4	1.49%
10.129.102.20	4	1.49%
10.129.102.2	4	1.49%
10.129.102.19	4	1.49%
10.129.102.18	4	1.49%
10.129.102.17	4	1.49%
10.129.102.16	4	1.49%
10.129.102.15	4	1.49%
10.129.102.14	4	1.49%
10.129.102.13	4	1.49%
10.129.102.12	4	1.49%
10.129.102.11	4	1.49%
10.129.102.10	4	1.49%
10.129.102.1	4	1.49%
10.129.102.0	4	1.49%
10.129.56.6	3	1.12%
205.188.226.248	2	0.75%
10.129.102.31	2	0.75%
10.129.102.30	2	0.75%
10.129.102.29	2	0.75%
10.129.102.28	2	0.75%
10.129.102.27	2	0.75%
10.129.102.26	2	0.75%
10.129.102.25	2	0.75%
10.129.102.24	2	0.75%
10.129.102.23	2	0.75%

Table 4.1: IP addresses participated in the first scenario as sourc

Destination IP Addresses	Count	Percentage
10.129.211.13	126	47.01%
61.189.243.240	7	2.61%
216.234.235.165	3	1.12%
10.129.56.6	3	1.12%
10.25.102.9	2	0.75%
10.25.102.8	2	0.75%
10.25.102.7	2	0.75%
10.25.102.6	2	0.75%
10.25.102.5	2	0.75%
10.25.102.4	2	0.75%
10.25.102.31	2	0.75%
10.25.102.30	2	0.75%
10.25.102.3	2	0.75%
10.25.102.29	2	0.75%
10.25.102.28	2	0.75%
10.25.102.27	2	0.75%
10.25.102.26	2	0.75%
10.25.102.25	2	0.75%
10.25.102.24	2	0.75%
10.25.102.23	2	0.75%
10.25.102.22	2	0.75%
10.25.102.21	2	0.75%
10.25.102.20	2	0.75%
10.25.102.2	2	0.75%
10.25.102.19	2	0.75%
10.25.102.18	2	0.75%
10.25.102.17	2	0.75%
10.25.102.16	2	0.75%
10.25.102.15	2	0.75%
10.25.102.14	2	0.75%
10.25.102.13	2	0.75%
10.25.102.12	2	0.75%
10.25.102.11	2	0.75%
10.25.102.10	2	0.75%
10.25.102.1	2	0.75%
10.129.102.31	2	0.75%
10.129.102.30	2	0.75%
10.129.102.3	2	0.75%
10.129.102.29	2	0.75%
10.129.102.28	2	0.75%
10.129.102.27	2	0.75%
10.129.102.26	2	0.75%
10.129.102.25	2	0.75%
10.129.102.24	2	0.75%
10.129.102.23	2	0.75%
10.129.102.22	2	0.75%
10.129.102.21	2	0.75%
10.129.102.20	2	0.75%
10.129.102.2	2	0.75%
10.129.102.19	2	0.75%
10.129.102.18	2	0.75%
10.129.102.17	2	0.75%
10.129.102.16	2	0.75%
10.129.102.15	2	0.75%
10.129.102.14	2	0.75%
10.129.102.13	2	0.75%
10.129.102.12	2	0.75%
10.129.102.11	2	0.75%
10.129.102.10	2	0.75%
10.129.102.1	2	0.75%
10.129.102.0	2	0.75%
205.188.226.248	1	0.37%
10.25.102.0	2	0.75%
10.129.102.9	2	0.75%
10.129.102.8	2	0.75%
10.129.102.7	2	0.75%
10.129.102.6	2	0.75%
10.129.102.5	2	0.75%
10.129.102.4	2	0.75%

Table 4.2: IP addresses participated in the first scenario as destinations

4.3.3 Stages of the attack

It has been found in the trace file that the first packet indicates that the IP identified earlier (10.129.211.13) carried out a DNS query to a domain name (bbjj.household.com) as shown in Table 4.3. This operation looks absolutely normal and does not give any indication of a problem. Although the communication seems legitimate, it has been found with deeper analysis that the DNS server that was queried is one on the DNS blacklist.

No.	Time	Source	Destination	Protocol	length	Info
1	0	10.129.211.13	10.129.56.6	DNS	77	Standard query 0x0006 A bbjj.household.com

Table 4.3: DNS Query

10.129.211.13 then received a reply (DNS response). The DNS response received gives another indication of irregular behaviour as it contains eleven IP addresses as shown in Table 4.4 while a normal DNS response contains 5 IP addresses.

No.	Time	Source	Destination	Protocol	Length	Info
2	0.237997	10.129.56.6	10.129.211.13	DNS	399	Standard query response 0x0006 CNAME ypgw.wallloan.com A 216.234.235.165 A 151.198.6.55 A 216.234.247.191 A 68.112.229.228 A 61.189.243.240 A 218.12.94.58 A 61.145.119.63 A 202.98.223.87 A 218.249.83.118 A 68.186.110.158 A 221.208.154.214

Table 4.4: DNS Response

The compromised host then tries to establish a connection with some of the IP addresses returned in the DNS response. Those IPs

responded by ICMP messages, as shown in table 4.5, to say that the destination is unreachable.

No.	Time	Source	Destination	Protocol	length	info
3	0.239858	10.129.211.13	216.234.235.165	TCP	62	1047->18067 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1
4	0.240407	216.234.235.165	10.129.211.13	ICMP	70	Destination unreachable (Port unreachable)

Table 4.5: Failure to Establish a Connection

It has been found that the compromised host sent another DNS query targeting the canonical name (ypgw.wallloan.com) found in the DNS response on the second packet. A DNS response is then returned containing eleven IP addresses. The compromised host tried then to establish a connection with one of the IP addresses returned and it succeeded to establish a connection with 61.189.243.240 as shown in table 4.6.

No.	Time	Source	Destination	Protocol	length	info
11	337.7635	10.129.211.13	61.189.243.240	TCP	62	1048->18067 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1
12	338.1601	61.189.243.240	10.129.211.13	TCP	62	18067->1048 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 SACK_PERM=1
13	338.1603	10.129.211.13	61.189.243.240	TCP	54	1048->18067 [ACK] Seq=1 Ack=1 Win=64240 Len=0

Table 4.6: Failure to Establish a Connection

The compromised host then started to send packets to the targeted host as shown in Table 4.7. By looking at the contents of the conversations between the compromised and targeted hosts, it turned out that it contains commands used by botnet as shown in table 4.8.

No.	Time	Source	Destination	Protocol	length	info
14	338.1604	10.129.211.13	61.189.243.240	TCP	67	1048->18067 [PSH, ACK] Seq=1 Ack=1 Win=64240 Len=13
15	338.7196	61.189.243.240	10.129.211.13	TCP	60	18067->1048 [ACK] Seq=1 Ack=14 Win=65522 Len=0
16	338.7196	10.129.211.13	61.189.243.240	TCP	71	1048->18067 [PSH, ACK] Seq=14 Ack=1 Win=64240 Len=17
17	339.1223	61.189.243.240	10.129.211.13	TCP	77	18067->1048 [PSH, ACK] Seq=1 Ack=31 Win=65505 Len=23
18	339.1224	10.129.211.13	61.189.243.240	TCP	75	1048->18067 [PSH, ACK] Seq=31 Ack=24 Win=64217 Len=21
19	339.6067	61.189.243.240	10.129.211.13	TCP	110	18067->1048 [PSH, ACK] Seq=24 Ack=52 Win=65484 Len=56
20	339.6068	10.129.211.13	61.189.243.240	TCP	72	1048->18067 [PSH, ACK] Seq=52 Ack=80 Win=64161 Len=18
21	340.0053	61.189.243.240	10.129.211.13	TCP	257	18067->1048 [PSH, ACK] Seq=80 Ack=70 Win=65466 Len=203

Table 4.7: Communication between the compromised and targeted host

Packet No.	Command
14	USeR l l l l
15	
16	NiCK p8-00196671
17	:a7 001 p8-00196671 :
18	USeRHOST p8-00196671
19	:a7 302 p8-00196671 :p8-00196671=+l@010.129.211.13
20	JOiN #p8 ihodc9hi
21	:a7 332 p8-00196671 #p8 :!Q gfcagihehehadkcpcpgigpgngfhegphhgocogbgpgmcogdgpgncphihihigmgpgmhh hegggjgigbhihihihicphdgpgdglhddjgbcogkhagh :a7 333 p8-00196671 #p8 a 1134159047 :a7 366 p8-00196671 #p8 :

Table 4.8: Bot net commands used between the compromised and targeted hosts

4.3.4 Summary of the Scenario

This scenario gives an example of how attackers can register machines to its bot army. Figure 4.1 gives an overview of the sequence of the attack. The figure shows that the attacker used the compromised host to contact a bad DNS server. The DNS server returned an unusual DNS response containing 11 IP addresses, while a normal response normally does not return more than five IP addresses. The attacker used the compromised host to scan IP addresses returned in the DNS query response and tried to establish communication with them. After a successful 3-handshake with one of the IP addresses returned in the response, the attacker sent packets that contain commands used by the botnet.

4.3.5 Analysis Outcome:

Some steps in this scenario could be considered to predict the occurrence of the attack. Detecting a DNS query with a bad DNS server can trigger an alert of malicious traffic. In addition, an irregular DNS response can indicate unusual behaviour. Moreover, sending packets containing commands used by botnet gives a strong indication that the traffic is malicious.

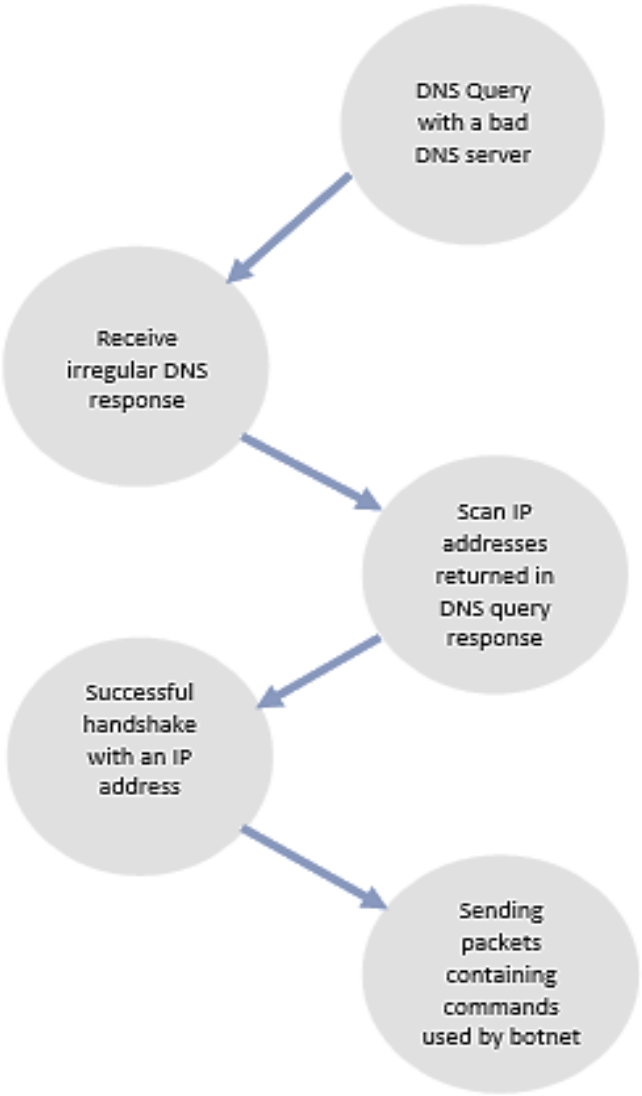

Figure 4.1: Stages of scenario A

4.4 Scenario B

4.4.1 Social Engineering

This scenario shows how social engineering can play a role in constructing a multi-stage attack. Social Engineering is defined by Chitrey (2012) in his comprehensive study of social engineering-based attacks as follow:

"Social Engineering is the art of exploiting the weakest link of information security systems: the people who are using them. Social Engineering is a method of gathering information and performing attacks against Information and Information Systems."

In other words, it is the art of abusing human behaviour in order to violate security without victims realising that they have been manipulated (SANS, 2007). Another comment added by Mitnick in an interview with the BBC News Online (2002) shows the role of social engineering in constructing attacks:

"What I found personally to be true was that it's easier to manipulate people rather than technology. Most of the time organisations overlook that human element."

4.4.2 Operation Shady Rat Attack

One of the multi-stage attacks that is social engineering-based is Operation Shady Rat. This attack was categorized by MacAfee (2011) as an advanced persistent threat. In addition, they describe it as one of the largest series of cyber-attacks ever. This attack started in 2006 and was reported in 2011 as hitting more than 72 large organisations, including twenty two government organisations, thirteen defence contractors, ten technology and electronic firms, eight policy influencers, and five 2008 Olympics related organisations (Talglobal, 2011). The next section will show how this attack can target an organisation.

An Operation Shady Rat attack involves five steps. In the first step, attackers select one or more organisations, then email individuals who work at those organisations. The emails sent contain information that attracts those individuals. Those emails also contain attached files that are relevant to the email body. Those files appear to recipients as normal files such as Word, Excel, or pdf files, but they are loaded with malicious code. For example, employees in a marketing company have a high interest in getting new contacts. Therefore, attackers may target this group by sending an email attached with an Excel file containing a contacts list.

In the second stage, recipients download the attached files, then open them. At the point of opening the file, the malware is installed on the victim's computer, thus compromising their computer.

In the third stage, the installed malicious program tries to establish a connection with a remote site specified in the code. The remote site URL does not look suspicious and it looks like a link to an image or normal html file, but the returned contents from that URL contains some information used by the malicious code. That information cannot be seen as being suspicious content, as it appears as a part of the html content. In addition, that information may be encoded or encrypted, so it will be difficult to analyse. For example, html comments can be used to embed the information that malware uses inside the html content. The comments are visible to end users, look absolutely legitimate, and cannot be seen as any kind of threat. The html comments may contain an IP address of a remote server or a command encoded in an encrypted or encoded format as shown in figure 4.2.

```
<!-- {5e1468jhsaa3q} -->
<!-- {8wfd2f7il2xfh} -->
<!-- {yaqwehd761mnb} -->
<!-- {yaqwehd761mnb} -->
<!-- {UGw^ddd,wddaa} -->
<!-- {z2x^4r2,aqwrd} -->
<!-- {saw^dwa,1jssa} -->
```

Figure 4.2: Example of HTML comments used embedded in HTML to be used by Malware

In the fourth stage, the installed malicious code establishes a connection with the IP address obtained in the third stage. In the fifth stage, attackers at the remote site establish a remote shell and run shell commands targeting the compromised machine. Attackers at this point can upload or download from the compromised side. Figure 4.3 shows the sequence of this scenario.

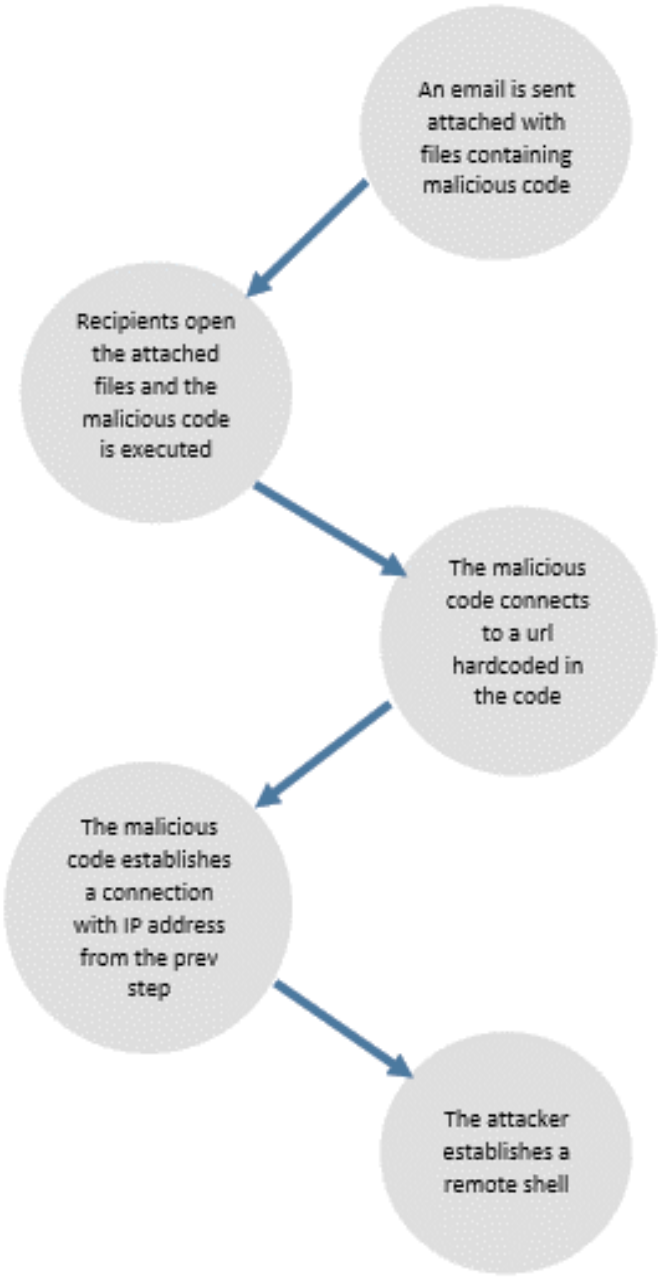

Figure 4.3: Stages of scenario B

4.4.3 Analysis Outcome

All steps of this scenario look legitimate and not suspicious. The only thing that can be checked, that may give an indication about suspicious traffic between the malicious code and other servers, are the reputations of the servers. There are some web services available and updated on a daily

basis, that provide reputation and more information about servers involved by passing URLs or IP addresses. Based on an evaluation obtained from those web services, the system can raise an alert of potential attacks.

4.5 Scenario C

4.5.1 CRLF Injection

The CRLF (Carriage Return and Line Feed) injection, which is also known as HTTP Response splitting, is an attack that can be easily constructed. However, it is an extremely destructive web attack. Attackers construct this kind of attack by exploiting vulnerable web applications that may allow also other types of vulnerabilities, such as cross site scripting and cross site forgery. The CRLF injection is carried out by injecting a very significant sequence of characters into web requests. This sequence contains two special characters representing EOL (End of line), which is used as a marker for many protocols, including such as HTTP and NNTP. In web applications, headers are split-based on the position of CRLF in requests. Malicious users inject their own CRLF sequence into an HTTP request. In the absence of filtering malicious inputs, malicious users will be able to control the functionalities of a web application function. In the next section, two examples of CRLF injection will be discussed showing how CRLF injections can be employed by attackers to construct multi-stage attacks (Hall, 2011).

4.5.2 Scenario C.1

This scenario is based on exploiting an insecure web application. This insecure web application can give a chance for attackers to get access to machines. The scenario shows how attackers exploit a vulnerable PHP web application to make a CRLF injection. The first step in this attack is carrying out a web vulnerability scan on a web server. This scan gives an attacker information about PHP configurations and different URLs,

including POST and GET parameters sent with them. The attacker then uses that information to send an email to a victim containing a CRLF-manipulated link. This link looks legitimate, but it contains parameters set to values that makes a vulnerable web application open a different URL rather than the specified URL in the code, as shown below:

Consider the PHP script below is saved as getfile.php:

```
<?php
        $folder = $_GET['folder'];
        $file = $_GET['file'];
        passthru("http://www.sitea.com/api?folder=$folder&file=$file");
?>
```

If an attacker tries to send send an email containing a link similar to:
getfile.php?folder=visby&file=gotland%20HTTP/1.0%0D%0AHost%3A%20www.siteb.com%0D%0AUser-Agent%3A%20Ulf/0.0%0D%0AReferer%3A%20http%3A%2F%2Fwww.gnuheter.org%2F%0D%0ACookie%3A%20user%3Dulf%0D%0A%0D%0A
(should be on one line)
This HTTP query will be sent to www.site1.st:

```
GET /api?folder=visby&file=gotland HTTP/1.0
Host: www.siteb.com
User-Agent: Ulf/0.0
Referer: http://www.gnuheter.org/
Cookie: user=ulf

HTTP/1.0
Host: www.sitea.com
User-Agent: PHP/4.1.2
```

As you can see, the real headers from PHP are sent as well, but the web server ignores them, as we send two CRLFs before them to indicate that the headers are over.

Figure 4.4: CRLF Injection on a PHP script

The injected URL may point to a file that runs on the victim's machine to push a remote shell for the attacker. The attacker proceeds by getting access to the web server, then downloads files or scans the network to find information they are interested in, or find targets they want to destroy. Figure 4.5 shows the different steps that occur during this attack.

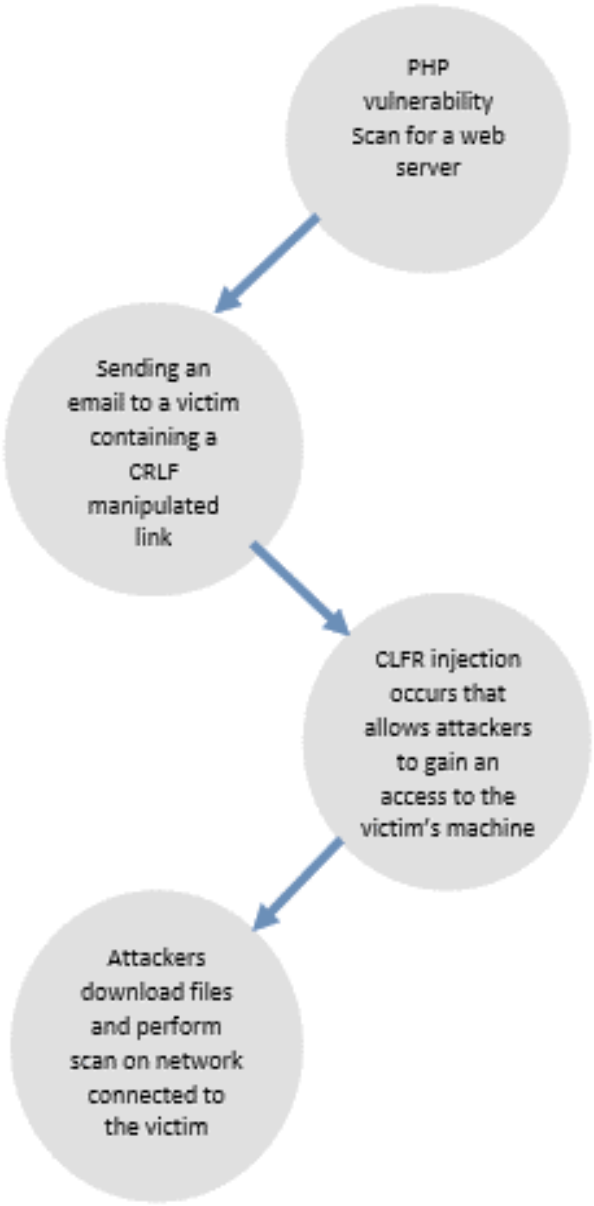

Figure 4.5: Stages of scenario C.1

4.5.3 Scenario C.2

This scenario is similar to the above; the difference in this scenario is that attackers injects html contents that will be displayed to the victim as follows:

http://www.site1.com/login.php?param1=%0d%0aContent-Type: text/html%0d%0aHTTP/1.1200OK%0d%0aContent-Type: text/html%0d%0a%0d%0a%3Chtml%3ELoginContent%3C/html%3E

When a victim receives an email that contains a manipulated link similar to the one above, a login page will be displayed similar to the one displayed on the original website. The victim may at this point enter their login details that will be sent later to attackers, rather than the host server. Attackers then use the login details to steal valuable information.

4.5.4 Analysis Outcome

This type of attack can be predicted or stopped at different points. The first point is checking parameters sent with web requests coming to the web server, whether it can cause CRLF injections or not. In addition to that, outgoing requests from the web server can be checked to see whether they go to trusted destinations or not.

4.6 Scenario D

4.6.1 Vulnerable FTP Service

File transfer protocol (FTP) is widely and commonly used by many organisations, to transfer files over the internet, due to its simplicity. However, there are some design decisions in that protocol that can be exploited by a malicious user (Lindfors, Peuhkuri, 1999). The next section gives an example of how an attacker can obtain unauthorised access through a vulnerable FTP service.

4.6.2 Scenario Description

This scenario shows how a malicious user attempts to obtain unauthorised access on an account on a local machine, which is part of a corporate network holding a server connected to the internet. That server runs a web service and a vulnerable FTP service. The first step the attacker

carries out is performing a port scan on externally visible IP addresses, using an Nmap security scanning tool. The aim of that scan is finding an open port in the targeted network. Figure 4.6 shows how the Nmap tool is used to find an open port.

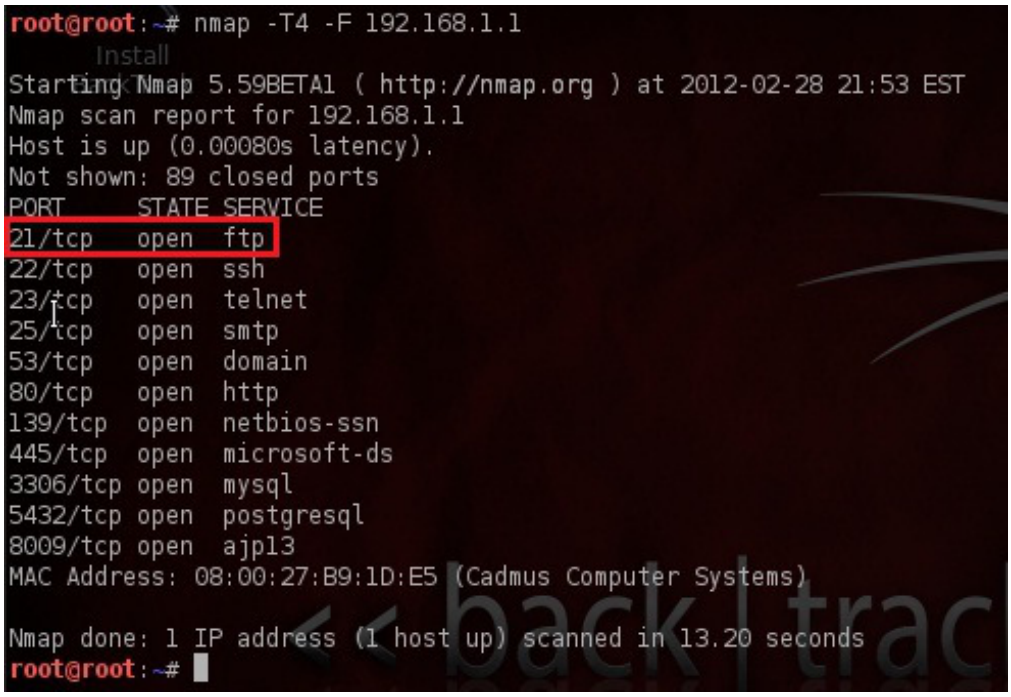

Figure 4.6: Using Nmap tool to find an open port (Pentration Testing Lab, 2012)

The attacker then tries to find a valid user name and password through a tool, such as THC Hydara or metasploit. Figure 4.7 shows how metasploit can be used to find a valid username and password for the vulnerable FTP service.

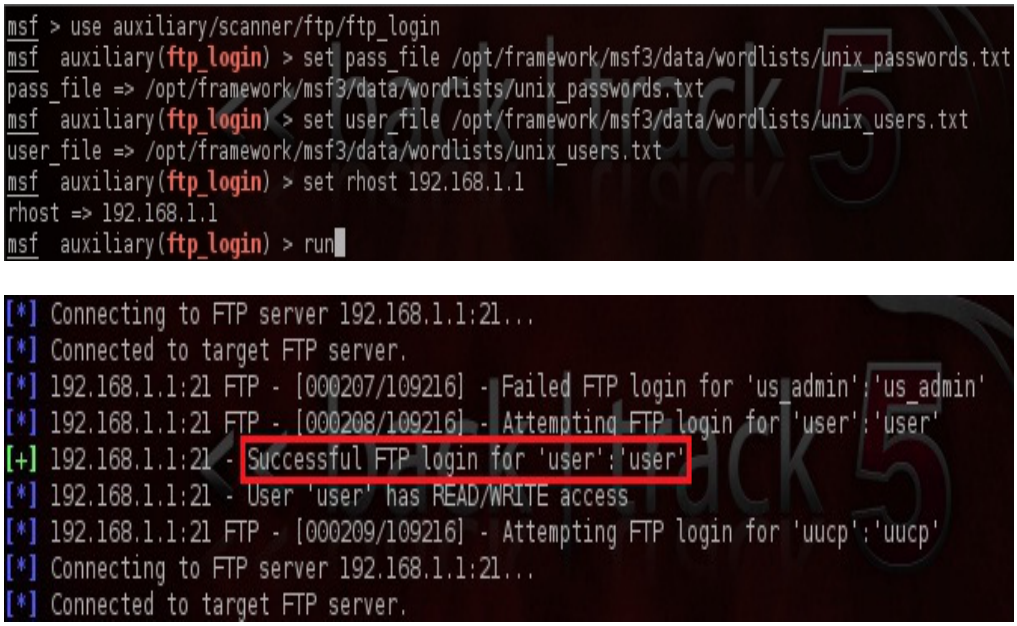

Figure 4.7: Using the metasploit tool to find a valid ftp login (Pentration Testing Lab, 2012)

The attacker gets access to the FTP server, using the login obtained in the previous step, then downloads or deletes files in that server. That scenario occurred in a large Dutch hospital, the Groene Hart Ziekenhuis, as reported by Spadaro (2013). It was found that the medical records of at least 50 patients were illegally accessed. Figure 4.8 shows the steps that attack goes through.

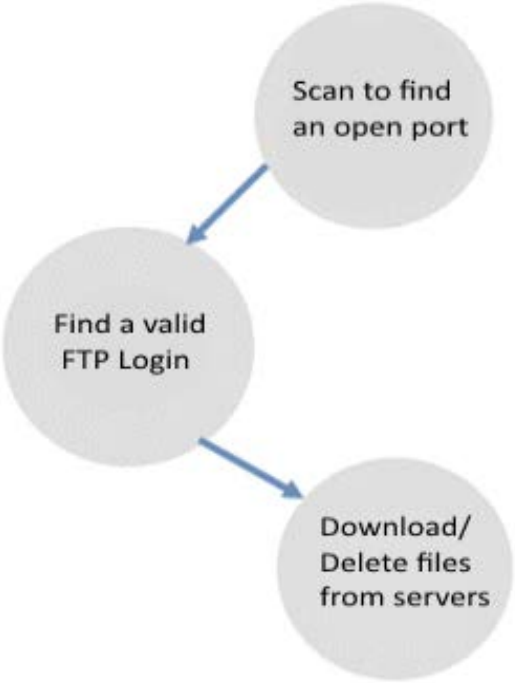

Figure 4.8: Stages of scenario D

4.6.3 Analysis Outcome

This attack can be predicted or stopped at the scanning point; a block scan from unauthorised IP addresses is needed. It can also be stopped when detecting upload/download from IP addresses with a low reputation.

4.7 Conclusion

In conclusion, it has been found that predicting or detecting multi-stage attacks is difficult to achieve through signature based solutions. Four different multi-stage attack scenarios have been analysed to understand the behaviour of multi-stage attacks. In the first scenario, a DNS query to a black-listed domain name gave a strong indication about malicious behaviour. The second scenario shows the sequence of Shady Rat Operation. The third scenario shows how attackers can exploit vulnerable web applications to construct an attack based on CRLF injections that is also known as header splitting. The last scenario shows how an attacker exploited a vulnerable FTP service to attack the network connected to that

service. It has been found that each stage in those scenarios looks like normal traffic, and does not violate any rule. It has also found that predicting those attacks may be achieved by carrying out a reputation check of IP addresses found in incoming and outgoing traffic. The next chapter will discuss the proposed solution to predict multi-stage attacks based on an IP reputation check.

CHAPTER 5

MULTI STAGE ATTACKS PREDICTION

5.1 Introduction

In the previous chapter, different multi-stage attack scenarios were discussed and analysed. It was found that each step in those scenarios tended to look innocent and was therefore difficult to capture as illegal traffic. Different solutions have been introduced to detect multi-stage attacks, some of those being event correlation-based. Event correlation-based solutions try to match network events with certain attack patterns. When a stream of network events matches a certain pattern, attacks can be stopped before progressing to the next stages. Many researchers claim the effectiveness of that approach in detecting multi- stage attacks. In a study by Spadaro (2013), an investigation was conducted to find out the relation between incidents and events that were monitored within today's IT infrastructures of large organisations. Another study was by Chen et al. (2006), who built a module called active event correlation on top of the bro network intrusion detection systems (NIDS).

Although being effective, this approach requires having up-to-date multi-stage attack patterns, which is not easy to achieve in a very short time, as discovering new complex attacks normally takes some time. The Shady Rat Operation attack is a good example of that; it started in 2006 but was only discovered in 2011.Thus, it has been decided to follow a different approach in this research, rather than network events correlation when proposing a solution for predicting multi-stage attacks. The following approach is based on evaluating the reputation of IP addresses participating in network traffic. Based on the evaluation, it can be decided whether we need to stop the traffic with evaluated IP addresses to block potential attacks. This chapter goes through the proposed solution that follows the latter approach; it consists of five sections. The first one gives an overview of the proposed solution, showing the different modules and flow of data. The second section discusses the first component in the solution, which is the network sniffer. The third section explains how the second component works to get information about IP addresses. The fourth section goes through the last module that is fuzzy logic-based. The fifth section shows how message brokers can be used to improve the performance of the proposed solution. The last section concludes this chapter.

5.2 An overview of the proposed solution

As mentioned in the previous section, the proposed solution is based on evaluating the reputation of IP addresses participating in the captured network traffic. The solution consists mainly of three modules as shown in figure 5.1. The first module (Network Sniffer) is responsible for monitoring network traffic by reading incoming and outgoing traffic. This module extracts IP addresses found in network packets; it reads then passes them to the next module (IP info finder). The IP info finder is responsible for finding information related to the IP addresses. The information obtained by the second module includes IP geographic information and other information that shows whether the IP addresses to be checked are

malicious. The last module in the proposed solution is fuzzy logic-based; fuzzy logic has been chosen rather than other data mining techniques, due to its effectiveness in dealing with uncertainty problems. It receives information obtained by the IP info finder to be processed through it. The output of this module will be in the form of a probability of having malicious network traffic. Based on the produced output from the fuzzy logic module, action will be taken. The action can be in the form of an email to administrators, or updating the firewall rules, to blocking communication with the discovered malicious IP addresses.

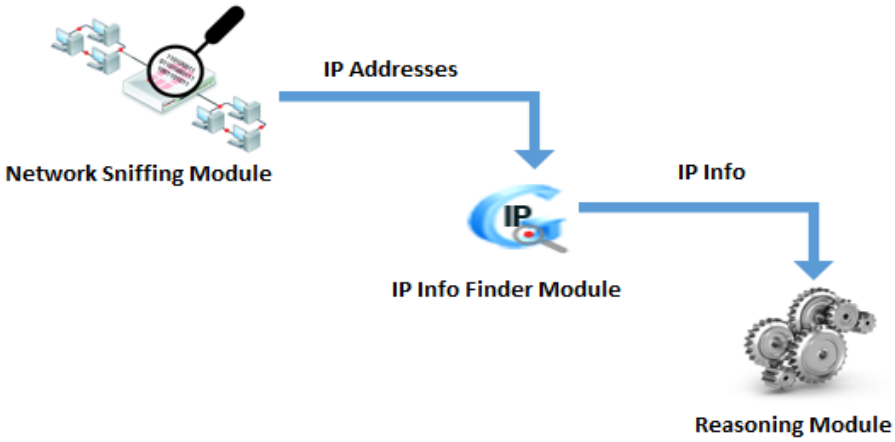

Figure 5.1: An overview of the proposed solution to detect multi-stage attacks

5.3 Network Sniffing Module

5.3.1 Choosing a sniffing tool

The network sniffing module is responsible for monitoring the network traffic. In other words, it captures incoming and outgoing network traffic. There are many available sniffing tools that can do this job, and these tools can be either hardware or software. Three parameters have been considered when choosing a sniffing tool for the solution proposed in this study. The

first one is the portability; it is necessary to use a sniffing tool that works with different operating systems (e.g. windows, UNIX). The second parameter to be considered is the simplicity to integrate into the solution; it will be difficult to integrate a sniffing tool into the solution, if it is a command line or provides APIs. The third parameter is the simplicity in obtaining information from the captured data.

Considering those parameters when looking at snoop, it was found that it bundled on the Solaris operating system. However, there is a Linux and Windows versions of this tool. Snoop is also a command line interface. The main disadvantage of this tool is that it lacks the capability to reassemble IP fragments, as reported by So-In (2006). In addition, this tool produces the output in a text format and does not provide a graphical interface, that can help in conducting further network traffic analysis.

Another tool looked at is Microsoft Network Monitor; it is bundled with Microsoft Windows and it runs only on Windows NT Server 4.0, Windows 2000 Server, or Windows Server 2003 and does not have a distribution on any other operating system. It has a simple and friendly graphical interface and cannot be used through a command line interface. However, all functionalities provided through the graphical interface can be used through Network Monitor API.

The last tool looked at is TCPDUMP; it is mainly bundled with Linux but available for many operating systems such as Solaris and Mac Os X. It is also available for Windows as Windump. This tool is a command line tool and does not have a graphical interface. However, there is other software developed to present the output of this tool in a graphical format, such as wireshark.

Looking at table 5.1, that gives a comparative overview for the tools mentioned above, and based on the three parameters mentioned earlier, it was found that the TCPDUMP is the most suitable tool for monitoring network traffic in the proposed solution.

TCPDUMP will be used to read network traffic packet by packet, then extract IP addresses from the captured infomation. This can be achieved by using the following TCPDUMP command:

TCPDUMP -i <Network interface index> -c 1 –n

Three options have been used with the command; the first one (-i <Network interface index>) to specify the index of the network interface that will be monitored. The second one (-c 1) to specify the number of packets to be captured, in this case we specified this as one. The last option (-n) is used to show IP addresses of source and destination. The command can be modified to run in different operating systems, such as Windows, by modifying the call to the sniffing tool as follows:

pathtowindumpfolder/windump -i 2 -c 1 –n

Tool	Portability	Simplicity to integrate	Simplicity to obtain info
Snoop	Has distributions over many operating systems	A command line interface so it is easy to integrate	Does not have a graphical representation.
Microsoft Network Monitor	Runs only on Windows	Provides an API to simplify the integration	Has a simple graphical interface
TCPDUMP	Has distributions over many operating systems	A command line interface so it is easy to integrate	Can be used with software such as wireshark to obtain a graphical representation

Table 5.1: A comparative overview over different sniffing tools

Figure 5.2 shows what the output of the command looks like. The output shows that the source IP address is 192.168.0.2, while the destination IP address is 216.58.210.5.

```
TCPDUMP: listening on \Device\NPF_{F99E0F0C-CD4F-4139-AEAB-1A6B340FBE25}
20:30:26.810488 IP 192.168.0.2.62350 > 216.58.210.5.443: UDP, length 24
```

Figure 5.2: The output of the TCPDUMP command

The network sniffing module has been implemented, using a php script. The php script consists of an infinity loop. In each loop, the TCPDUMP command described above is executed then its output is parsed to extract IP addresses. The IP address is then passed to the next module (IP info finder). Figure 5.3 shows the implementation of the network sniffing module in PHP.

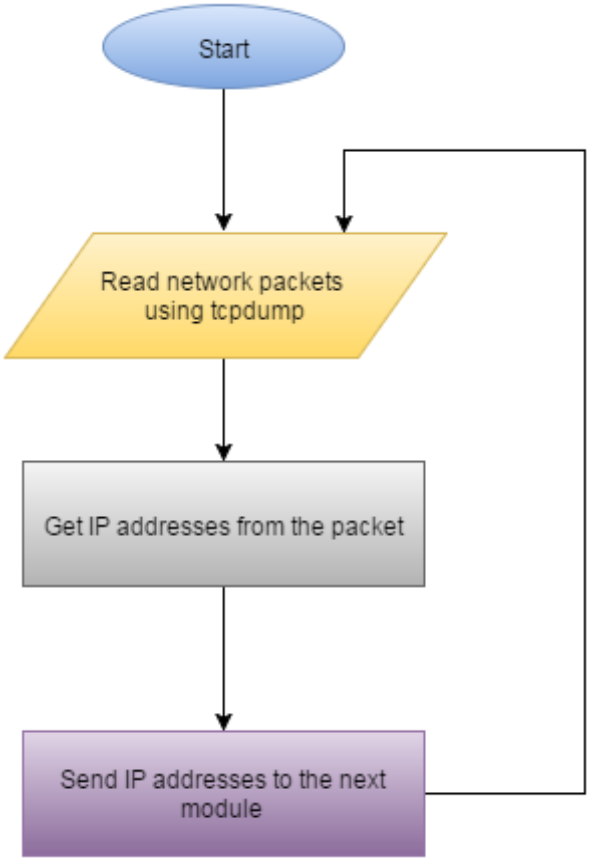

Figure 5.3: The flow chart of the network sniffing module

5.4 IP Information Finder Module

5.4.1 IP geographic Location

This module is responsible for finding information about IP addresses passed by the first module. The information gathered about those IP addresses will give a strong indication whether they can be source of malicious traffic or not. One element of the information being gathered is the IP geographic location. It was reported by Musthaler (2014) that 84% of malicious traffic in a recent quarter originated from ten countries. She also added that the attacker does not need to be in the country where the traffic has originated. In other words, they run their packets through compromised machines in those countries. Attackers direct traffic through open proxies in order to increase their threats. The traffic will appear as coming from many sources. That will give attackers the opportunity to overcome the problem of blocked traffic from a certain country, by directing traffic through a different country. It may be thought that the solution is to block traffic from countries with a high percentage of malicious traffic but this is not practical, as there may be legal traffic from these countries. Therefore, it has been decided not to consider all traffic coming from a black-listed country as malicious. However, geographic location will be considered as one of the parameters when evaluating IP addresses in the next module.

The proposed solution needs to know the countries black-listed by administrators, in order to identify whether the IP geographic location is suspicious or not. Thus, a simple user interface was developed (see Figure 5.4) to enable administrators to specify those countries; this list will be stored in a database.

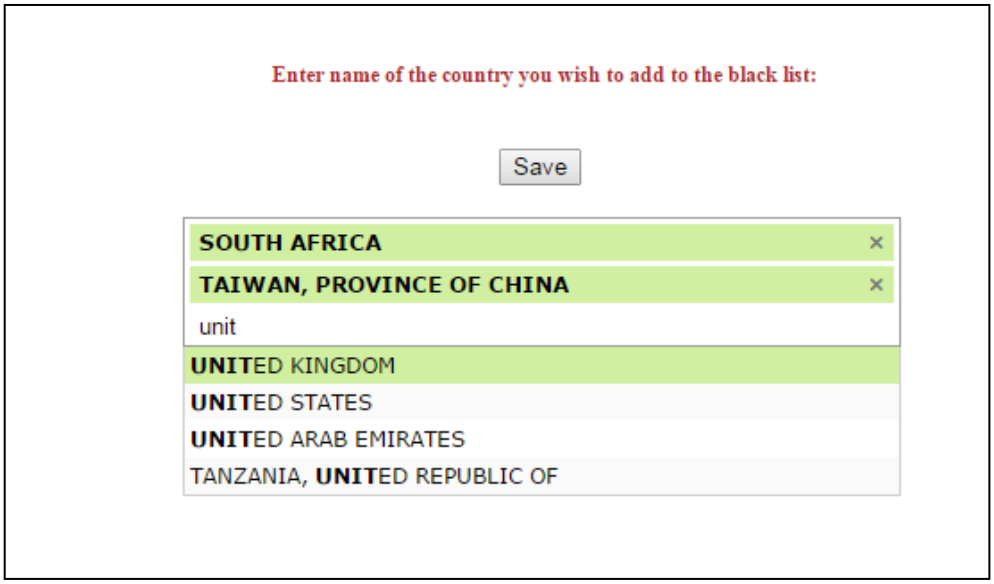

Figure 5.4: Black-Listed Countries Selection

The IP geographic location can be obtained by storing IP geographic information in a database, then checking the IP address against it to find its location. Another way to achieve this information is through one of the available web services (API). The main disadvantage of this first solution is the need to regularly update. On the other hand, there are many web services regularly updated and, therefore, it was decided to go with the second option. Neutrinoapi is one of those web services that provides a method (IP-info) for getting the IP geographic location. Table 5.2 shows the API request structure, while table 5.3 shows the API response.

Parameter	Required	Type	Default	Description
Ip	Yes	String		IPv4 address
reverse-lookup	No	Boolean	FALSE	Do reverse DNS (PTR) lookup. This option can add extra delay to the request so only use it if you need it

Table 5.2: API Request for Finding IP geographic location (Neutrino API, 2013)

Parameter	Type	Description
Valid	Boolean	Is this a valid IP address
Country	String	Full country name
country-code	String	ISO 2-letter country code
City	String	Full city name (if detectable)
Region	String	Full region name (if detectable)
Longitude	Float	Location longitude
Latitude	Float	Location latitude
Hostname	String	IP hostname (if reverse-lookup has been used)

Table 5.3: API Response for Finding IP geographic location (Neutrino API, 2013)

5.4.2 IP Block List

As mentioned earlier, IP geographic location is not the only criterion that can be used to judge whether IP addresses may be a source of malicious traffic. Thus, it is required to check other criteria in conjunction with the IP geographic location. One of those criteria is whether the IP address is on a block list or not. IP addresses that are on a block list can be spyware, hijacked, spam-bot, exploit-bot, bot, or flagged in Dshield. If one of those criteria is met, a flag of possible suspicious activity needs to be raised.

It will also be beneficial to check other criteria such as being an anonymous web proxy, or exit tor node; meeting only one of those criteria will not necessarily sound the alarm for potential suspicious activity. Neutrinoapi web services provide another method (IP-block list) to get information about those criteria for a specific IP address. Table 5.4 and 5.5 shows the API request and response structure for this method.

Parameter	Required	Type	Default	Description
Ip	Yes	string		An IPv4 address

Table 5.4: API Request to check whether IP is block listed (Neutrino API, 2013)

Parameter	Type	Description
is-listed	boolean	Is this IP on a blocklist
list-count	integer	The number of blocklists the IP is listed on
is-proxy	boolean	IP has been detected as an anonymous web proxy
is-tor	boolean	IP is coming from a TOR exit node
is-vpn	boolean	IP has been detected as coming from a VPN hosting provider
is-spyware	boolean	IP is being used for spyware, malware, botnets or other malicious activities
is-dshield	boolean	IP has been flagged on DShield (dshield.org)
is-hijacked	Boolean	IP is listed as being stolen or hijacked from the rightful address owner
is-spider	boolean	IP is a web spider or crawler (legitimate or otherwise)
is-bot	boolean	IP is hosting a malicious bot or is part of a botnet
is-spam-bot	boolean	IP address is hosting a spam bot, comment spamming or other spamming software
is-exploit-bot	boolean	IP is hosting an exploit finding bot or exploit scanning software

Table 5.5: API Response check whether IP is block listed (Neutrino API, 2013)

5.4.3 IP Rating

In addition to checking whether the IP is on a block list or not, Neutrinoapi web services provides another method (host-reputation) that checks the IP rating in Domain Name System Block Lists (DNSBL). In other words, this method checks the host's reputation. Table 5.6 and 5.7 shows the API request and response structure for this method.

Parameter	Required	Type	Default	Description
Host	Yes	string		An IPv4 address or a domain name. If you supply a domain name it will be checked against the URI DNSBL list

Table 5.6: API Request for Finding IP rating (Neutrino API, 2013)

Parameter	Type	Description
is-listed	Boolean	Is this host blacklisted
list-count	Integer	The number of DNSBL's the host is listed on
Lists	Array	An array of objects for each DNSBL checked, with the following keys: is-listed - true if listed, false if not list-name - the name of the DNSBL list-host - the domain/hostname of the DNSBL list-rating - the list rating [1-3] with 1 being the best rating and 3 the lowest rating txt-record - the TXT record returned for this listing (if listed)

Table 5.7: API Response for Finding IP rating (Neutrino API, 2013)

5.4.4 Implementation

This module has been implemented using a PHP script; the script curl library to make the required API calls (See Appendix F). The

information obtained from the API calls will then be passed onto the next module. Figure 5.5 shows the flow chart of this module.

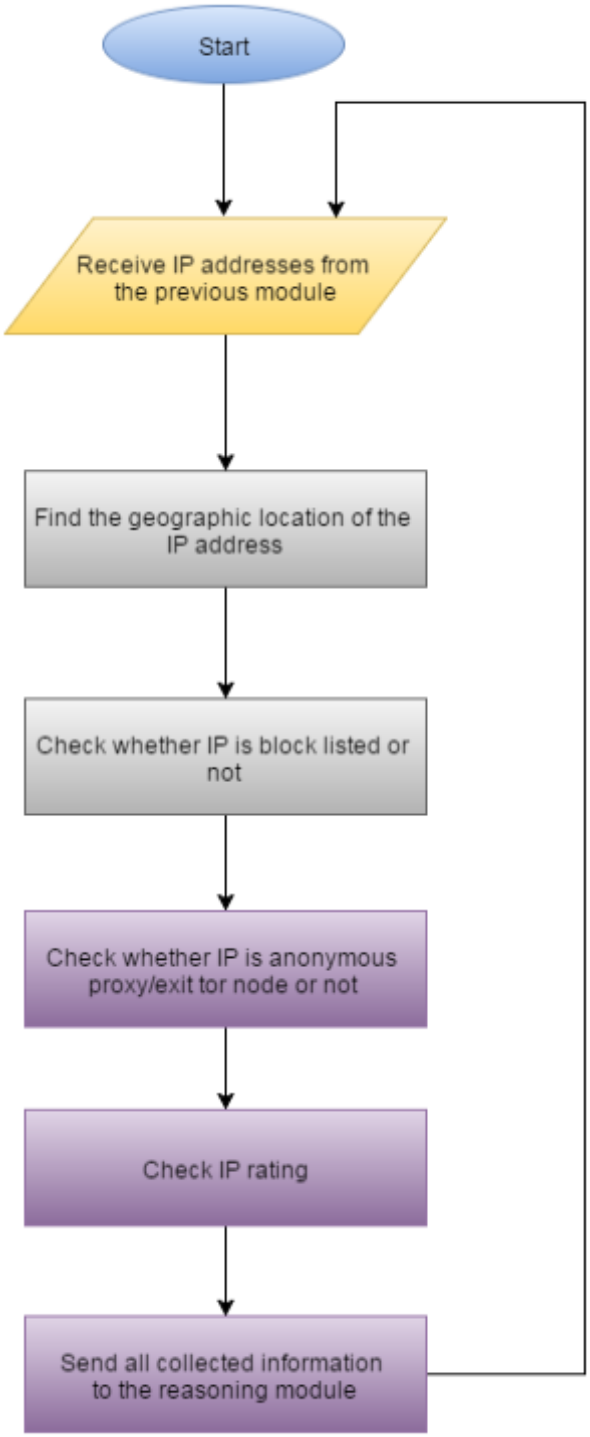

Figure 5.5: The flow chart of the IP info finder module

5.5 The reasoning module

5.5.1 Data Mining Technique Selection

The reasoning module is responsible for deciding whether there is possible malicious traffic from an IP address, based on data collected from the previous module. Analysing the collected data to give a decision can be achieved through one of the data mining techniques. It was decided to use the fuzzy logic in this module based on many reasons. The first one is that analysing the collected data can be simply modelled using the “if then" rules form, which is supported by fuzzy logic. In addition to this, there are some scenarios where there is no certainty for deciding whether an IP address is malicious or not. The fuzzy logic is suitable for those ambiguous scenarios (Albertos et al. 2008). Moreover, constructing the fuzzy rules for this system will not take much effort and time compared to machine learning algorithms. Machine learning algorithms require large data sets for training to obtain accurate results. In addition, the training time with a large data set is very time consuming. Another parameter to be considered in the choice of fuzzy logic is its simplicity to adapt to changes occurring in the reasoning model, as it requires only modification of the fuzzy rules. On the other hand, machine learning algorithm models need to be trained in that case. Pulo (1999), in his investigation about fuzzy logic and machine learning algorithms, supported the choice of using fuzzy logic rather than machine learning algorithms in such circumstances saying:

“Humans perform much better when they are able to interpret and gain meaning, understanding and information from the training data. This is when they can generalize best and draw the best conclusions. ML algorithms, however, have no such requirement, and can apply techniques such as decision trees and Bayesian inference to obtain results approaching the probabilistic optimum without any need to comprehend anything”

5.5.2 Pre-processing the inputs

The reasoning module receives its inputs from the IP info finder module; these inputs are shown in table 5.8.

Input name	Description
IP Geographic Location	Specifies which country the IP is based at
Is IP in a block list	Specfies whether the IP is found in a block list or not
Is IP an anynomous proxy	Specifies whether the IP is an anynomous proxy or not
Is IP a TOR exit node	Specifies whether the IP is a TOR exit node
IP Rating	An array that shows the IP rating on different DNSBL

Table 5.8: The reasoning module inputs

Some of the inputs described in the above table need to be pre-processed before applying them to the fuzzy logic as they are not in a format that can be handled by it. Table 5.9 shows how the inputs will be pre-processed.

Input name	Pre-processing rule
IP Geographic Location	The country name will be checked against the black listed country specified by the administrator. If it is found in that list, the value will be set to one. Otherwise, it will be set to zero. The input will be renamed to 'is IP in a black listed country'
Is IP in a block list	Does not need processing as it is a boolean value
Is IP an anynomous proxy	Does not need processing as it is a boolean value
Is IP a TOR exit node	Does not need processing as it is a boolean value
IP Rating	The average IP rating will be calculated. If the IP address is not found in any DNSBL, the value will be set to 3.

Table 5.9: Pre-processing the reasoning module inputs

5.5.3 Fuzzy logic

As described in chapter three, the fuzzy logic system consists of four elements; fuzzifier, rule base, defuzzifier and inference engine. These components, as shown in figure 5.6, interact with each other in order to produce an output. The following sections will discuss how each element will be used and configured in the reasoning module.

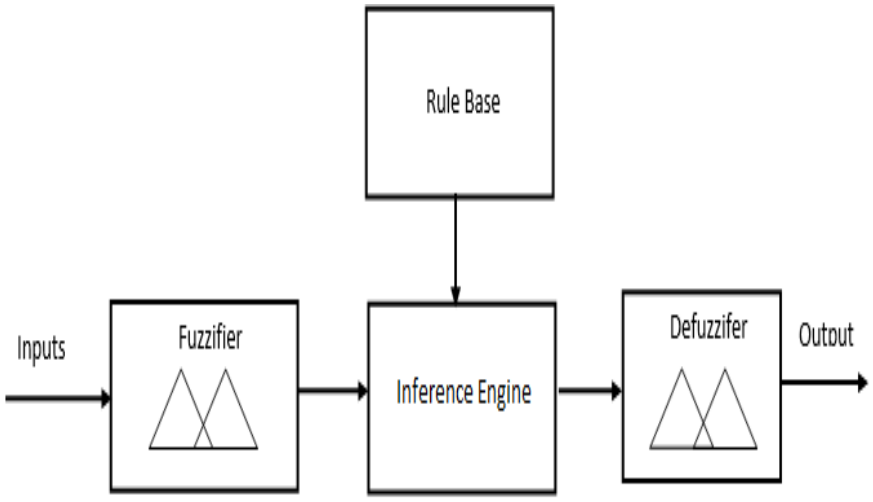

Figure 5.6: A fuzzy logic elements

The fuzzifier is responsible for converting the crisp inputs to fuzzy sets by using specified membership functions for each input. It was found that four inputs (is IP in black- listed country, is IP an anonymous proxy, is IP Tor Exit, and IP block listed) are Boolean, which can be handled by a singleton function. The membership function selected for each of these inputs, as shown in figure 5.7, is only set to one at a single value. On the other hand, the membership function selected for IP rating is specified using triangle functions, as shown in figure 5.8. The figure shows that IP reputation can be described as high or low in the selected membership function.

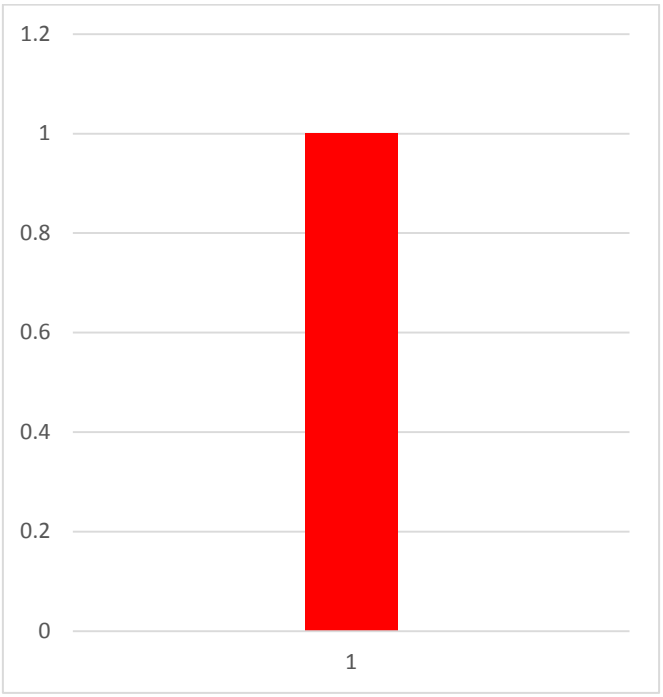

Figure 5.7: The membership function selected for the inputs having Boolean values

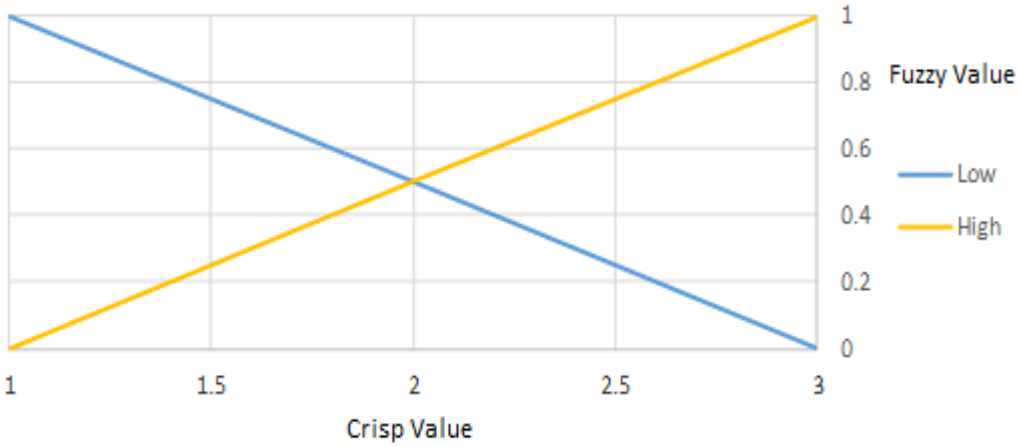

Figure 5.8: The selected membership function for IP reputation

Rule base is the part that contains the logic of producing the output. It contains a number of (if …. then) rules that will be used by the interference engine to produce a fuzzy output. Table 5.10 shows the rules used in the reasoning module to predict malicious traffic.

If condition	Then ststement
(IP in a block list)	Possible malicious traffic
(IP country in a black list) AND (IP is an anynomous proxy)	Possible malicious traffic
(IP country in a black list) AND (IP is a TOR exit node)	Possible malicious traffic
(IP Rating is low)	Possible malicious traffic

Table 5.10: If then rules used in the reasoning module

The first rule is straightforward, the IP will be considered as a malicious one if the IP address is found in a block list. Finding an IP in a block list means that the IP address has been reported to be used in malicious activities. The second and third rules check two parameters. One of them is whether IP is in the black listed countries or not. It is not practical to consider an IP as a malicious one if it is only located in one of the countries found in the black list as there may be legal traffic from those countries. Anonymous proxies and tor are used in a way that enable users to protect access the web anonymously. Attacker normally do not need to be in the listed countries, they direct their traffic through a proxy or tor located in one of those countries. Therefore, getting a traffic from anonymous proxies or tor exit nodes located in those countries raise an alert of potential malicious traffic. The last rule checks the average IP rating (the host reputation). The IP address will be considered malicious if the average rating is low.

The defuzzifier is responsible for converting the fuzzy output to a crisp value using a selected membership function for the output. Figure 5.9 shows the selected membership function for the output. The produced output gives the probability of having malicious traffic from the checked IP address. The final output will be considered as malicious if it is higher than 0.5, otherwise it will be considered as a normal.

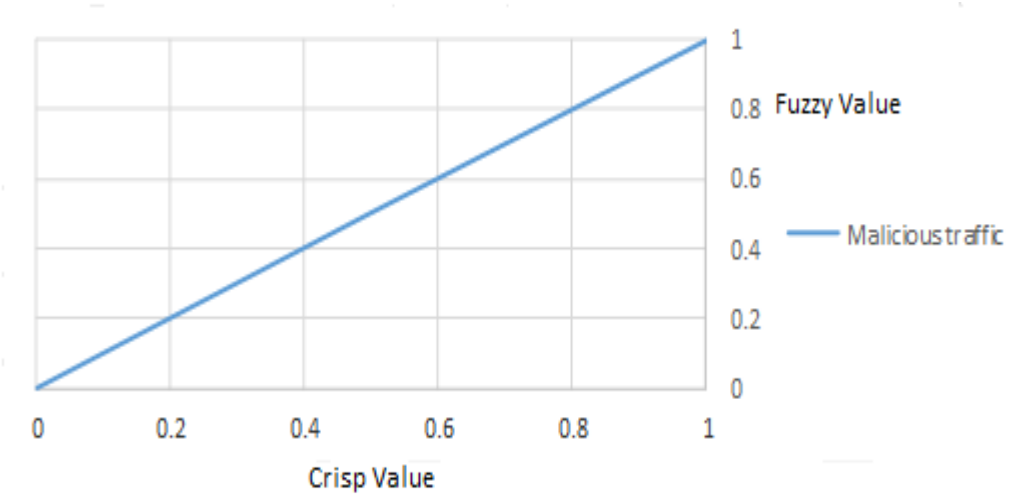

Figure 5.9: The selected membership function for the output

The inference engine can be considered as the heart of reasoning, as it is responsible for mapping given inputs to a fuzzy output, using the specified rules. The inference engine used in this module is mamdani, which is commonly used in fuzzy logic system and successfully applied in classification problems (Mathworks, 2015).

5.5.4 Implementation

The reasoning module has been implemented using PHP and MySQL. Figure 5.10 shows the flow chart of the reasoning module.

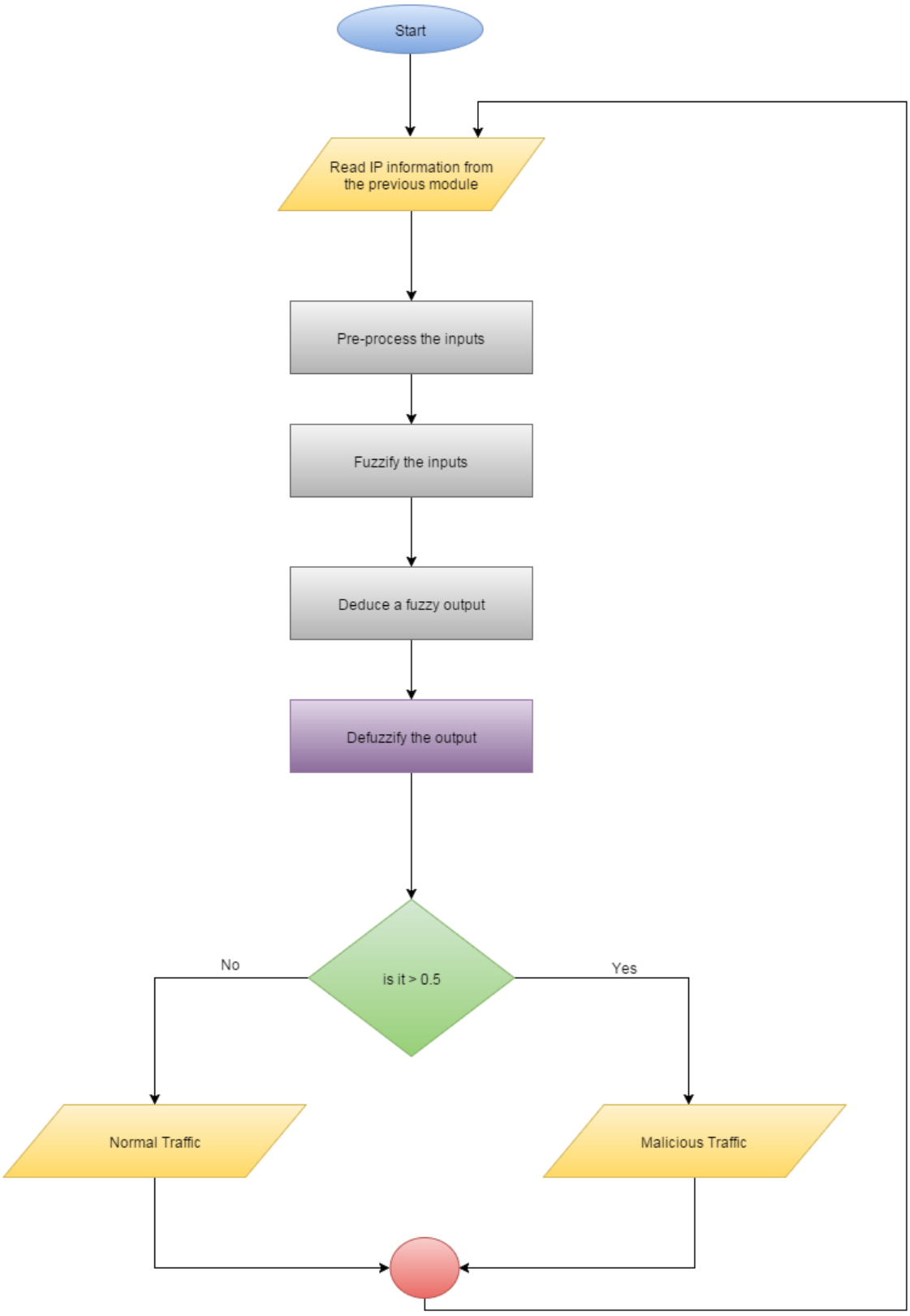

Figure 5.10: The flow chart of the reasoning module

5.5.5 Using Message Broker

It was found that evaluating each incoming or outgoing packet will take a certain time, as it involves sending API requests in addition to the processing time in the reasoning module. During that time, the system may send or receive many network packets that the proposed system may miss processing. Therefore, it is essential to modify the proposed system in a way that allows it to process all messages. One of the solutions that can solve this problem is to use a message broker between the network sniffing and IP info finder modules. The network sniffing module will act as a message producer that sends messages containing IP addresses to be checked; it will send a message once a network packet is received or sent, without the need to wait for other modules to finish their tasks. The messages sent by the network sniffing module will stay in a queue until one of the consumers can receive them for processing. The IP info finder will act as a consumer in this case. Figure 5.11 shows what the proposed solution will look like after using a message broker. The implementation of the network sniffing and IP info finder info will be slightly changed by using RabbitMQ library. RabbitMQ is one of message brokers widely used, and simple to use. Figure 5.12 shows how the flow chart of the network sniffing module after adding the message broker, while figure 5.13 shows the flow chart of IP info finder module as a consumer.

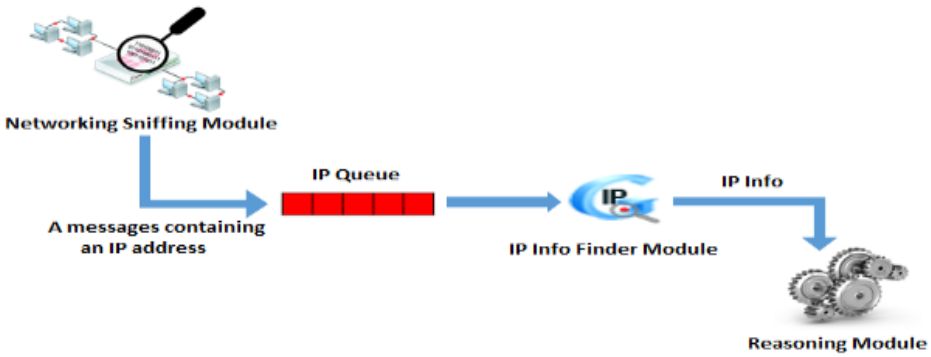

Figure 5.11: The modified version of the proposed solution after adding a message broker

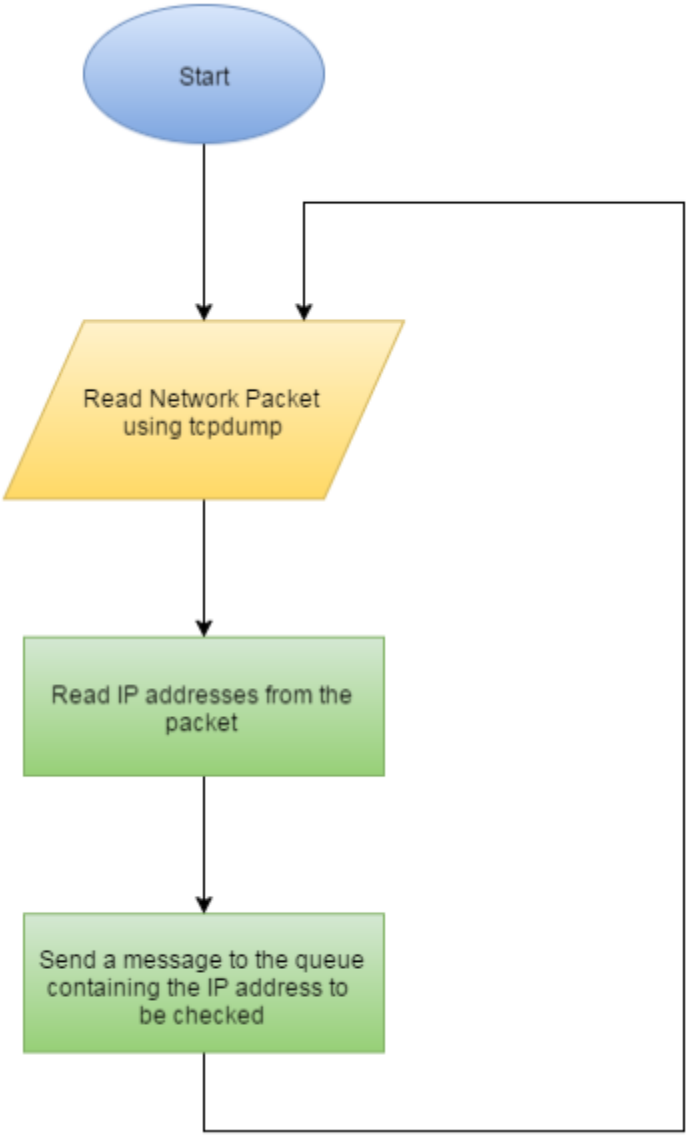

Figure 5.12: Network sniffing module when using message broker

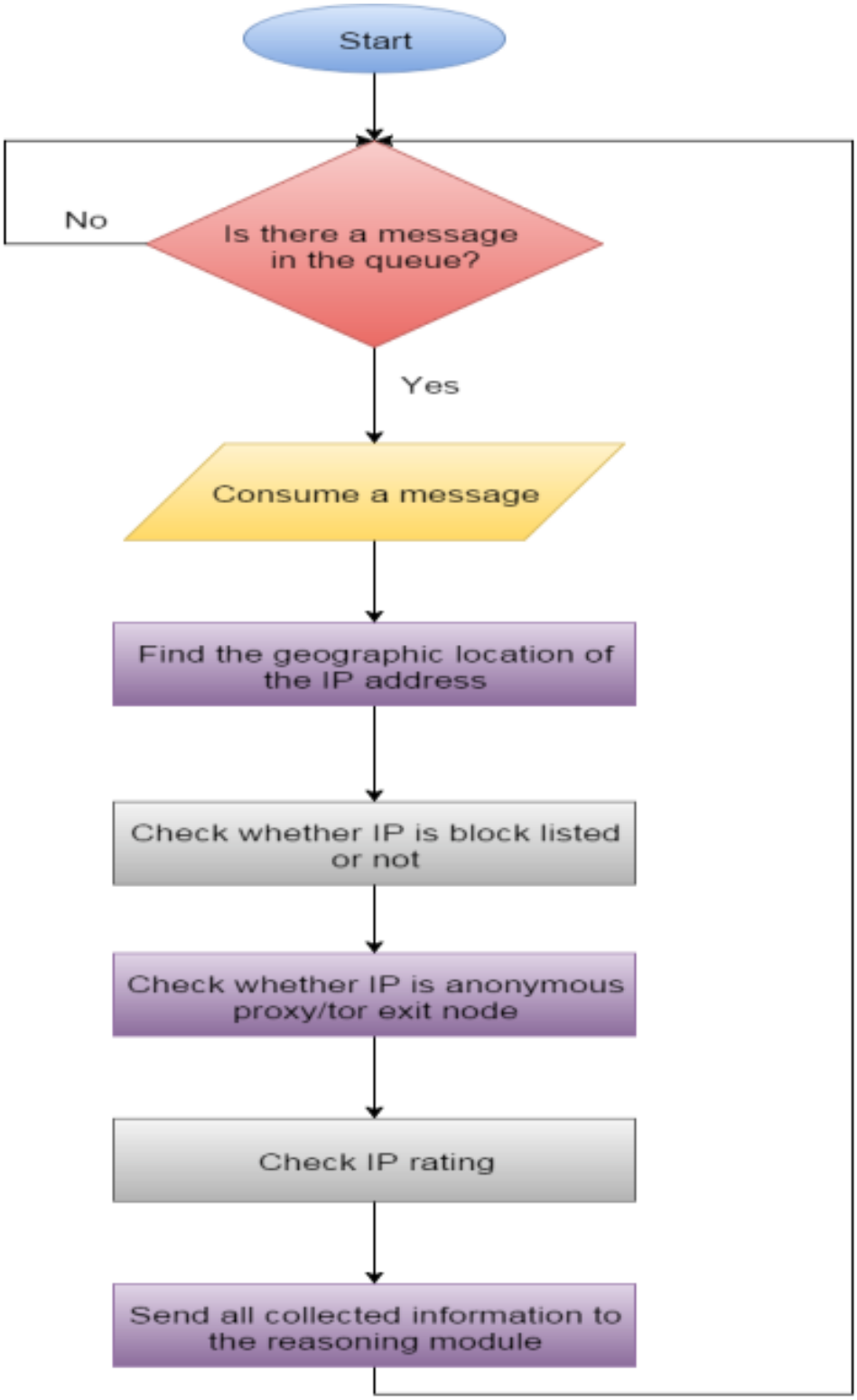

Figure 5.13: The flow chart of the IP information module when using message brokers

5.6 Summary

In conclusion, many researchers use the event correlation approach, but the downside of this approach is the need to keep the attack sequence patterns up-to-date, in order to detect multi-stage attacks. Thus, another approach was applied in this study, to predict multi-stage attacks. This approach depends on checking the IP addresses involved in network traffic. The proposed solution for using this approach consists of three modules. The first one (network sniffing) is responsible for reading network traffic, then extracting IP addresses from the packets. This module has been implemented by using the TCPDUMP tool. The second module (IP information finder) is responsible for getting information about IP addresses extracted at the initial stage. The information is checked by the second module, including the IP geographic location, and checking whether the IP is on a block list, whether the IP is an anonymous proxy, or a TOR exit node, and checking the IP rating in DNSBL. The last module (reasoning) is responsible for deciding whether IP addresses may be a source of malicious traffic or not, based on information passed from the second module. The outputs of the second module need to be pre-processed before processing them. The reasoning module was implemented using fuzzy logic. The reason for choosing the fuzzy logic rather than any of the machine learning algorithms is the nature of the problem; it can be simply solved using "if… then" rules. In addition, it requires less time and effort to adapt to changes in the reasoning logic compared to machine learning algorithms. Machine learning algorithms require large sets of training data to get accurate results and that consumes a lot of time in the training phase. One of the issues found in the proposed system is the high chance of missing some network packet, due to the time spent in obtaining the IP information and then processing in the reasoning module. This issue has been dealt with by using a message broker; the network sniffing module will queue IP addresses extracted from network packets, rather than passing them directly to the IP info module, and there is no need to wait for IP info finder and the

reasoning module to complete their job. The IP info module will then consume messages in the queue.

CHAPTER 6

EVALUATION

6.1 Introduction

In the previous chapter, the proposed solution for predicting multi stage attacks was discussed showing their different modules. This chapter evaluates the effectiveness of the proposed solution. The evaluation is divided into two stages. The first one is measuring the effectiveness of the solution by following a metrics based approach. This approach was introduced by Fink, et al. (2002). The approach looks at intrusion detection systems from different perspective; it includes logistics, architectural, and performance metrics. The logistic metrics allocates a score according to the perceived merit in each category in terms of maintainability, manageability, and dependency. The design metrics is used to find how well the system performs in terms of resources consumption and speed. The last metrics used in this approach is the confusion metrics (performance metrics), this metrics finds how well the system does its job (detecting multi stage attacks) in form of true positive, true negative, false positive, and false negative.

The second section of this chapter discusses the logistics metrics, the metrics includes evaluating distributed management, ease of configuration, ease of policy management, outsource solutions, and platform requirements. The third section looks at the design metrics that includes adjustable sensitivity, data storage, multi sensor support, firewall interaction, packet

loss, and system throughput. Each category in the logistic and design metrics will have a score between one and three (one is the lowest and three is the highest) based on number advantages and disadvantages For example, consider evaluating the system throughput. The system will score one If it has a low throughput while it will score two if it has a high throughput but with consuming a lot of hardware resources. On the other hand, the system will score three if it has a high throughput without consuming a lot of hardware resources. The fourth section provides a performance evaluation for the system in form of a confusion metrics. The last section gives a conclusion for this chapter.

6.2 Logistics Evaluation

6.2.1 Distributed Management

The distributed management for intrusion detection systems was described by Einwechter (2001) as:

"Multiple Intrusion Detection Systems (IDS) over a large network, all of which communicate with each other, or with a central server that facilitates advanced network monitoring, incident analysis, and instant attack data. By having these co-operative agents distributed across a network, incident analysts, network operations, and security personnel are able to get a broader view of what is occurring on their network as a whole."

By looking at the proposed solution, it has been found that it can support distributed management by having several network sniffing modules over a large network. The sniffing module will then queue messages that will be consumed by the IP info module that will then feed the reasoning module. Figure 6.1 shows the architecture of the solution when having a distributed management over a network. In this architecture, the centralized server will run both the IP information finder and reasoning

modules. The downside of supporting this structure is creating a bottle nick around the IP info and reasoning modules and slowing the process of detecting multi stage attack overall.

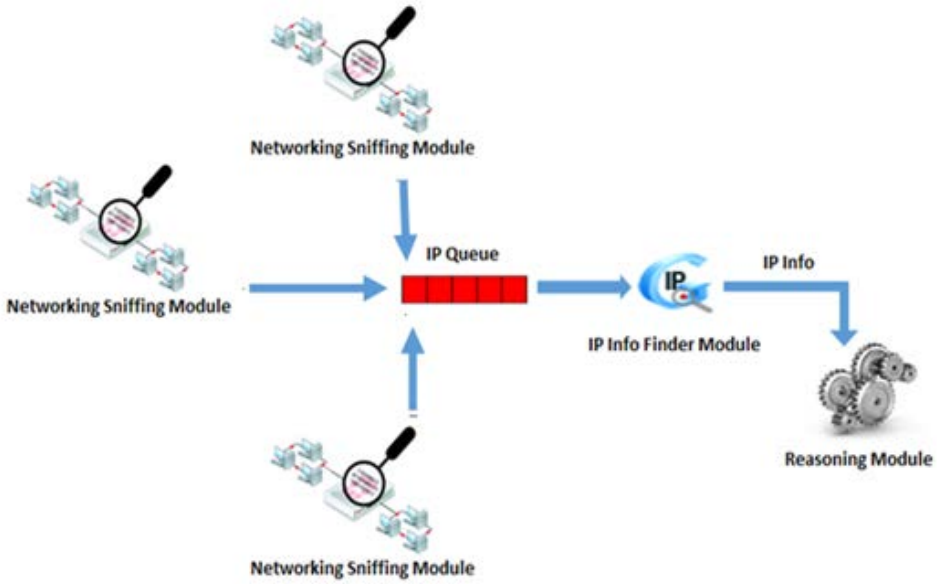

Figure 6.1: Distributed Management Architecture

6.2.2 Ease of Configuration

Ease of configuration means how easy to install and configure the system (Fink et al., 2002). In this context, the proposed system will be assessed in terms of how easy to install and configure its components. The system requires installing APACHE, PHP, MYSQL, TCPDUMP, and AMQP. The first four components are easy to install and configure. For example, installing APACHE on Linux environment requires running only a single command. Moreover, in platforms like windows APACHE, MYSQL, and PHP come in one package which simplifies the installation process. On the other hand, installing and configuring AMQP is not as simple as others components and passes through many steps to get running. Another point is considered as a disadvantage in this context is that configuring the components of the system is not centralized through one interface and a prior knowledge for each component is required in order to configure them.

6.2.3 Ease of Policy Management

Fink et al. (2002) describe measuring ease of policy management in their metrics approach as how easy to update, create, and manage detection rules. By looking at the proposed system, it has been found that detection rules are defined through the fuzzy rules. Updating or adding fuzzy rules is very simple and do not require changing the other reasoning module components. That advantage of the fuzzy logic was discussed in the previous chapter and it is one of the reasons of choosing it rather than any of machine learning techniques. However, this advantage does exist as long as there are no new inputs to be considered in deducing the output. It may be discovered later that there are more IP data (new inputs) indicates whether IP addresses are malicious or not. In addition to changing fuzzy rules, adding more inputs requires adding more membership functions to the fuzzifier.

6.2.4 Outsource Solutions

Measuring the level of dependency on external systems to run required services is one of parameters required to be highly considered when assessing a system. It has been found that the IP information module is highly dependent on an external system as it uses web services to get information about IP addresses. Although using web services to find IP information has the advantage of getting up to date information, it is considered at the same time as a disadvantage. Potts and Kopack (2003) listed the availability as one of web services pitfalls saying:

"Everyone who uses the Internet knows that no site is 100% available. It follows that Web services, which use the same infrastructure as Web sites, will not be 100% available either. Even if the server is up and running, your ISP might not be, or the ISP hosting the other side of the transaction might not be either."

They also mentioned that immutable interface is another issue with using web services as request and response structure may be changed in a

way that can break the IP information finder. Having the IP information finder in a faulty status means that the reasoning module will not be able to classify IP addresses as no inputs are provided by the IP information module in this case. As a result, malicious traffic may pass through the system without raising any alarm.

6.2.5 Platform Requirements

The system resources required to run the proposed system is another parameter needs to be looked at. The system requires APACHE, MYSQL, PHP, TCPDUMP and AMQP. All of those have distributions over different operating systems (MAC, Windows, Linux) which gives the system the flexibility to run on different platforms. In addition, all of those components are free. By looking at the hardware requirements, it has been found that there are no specific hardware requirements and it depends on traffic volume. For example, memory resources required for buffering messages in AMQP are specified based on traffic volume.

6.2.6 Conclusion

Table 6.1 shows the score for each item in the logistic assessment. The score for the distributed management item is two as the system supports it but with some potential issues in the buffering area. The score for of ease of configuration and policy management is two, many components can be easily installed but the configurations is not centralized on one user interface and scattered over different areas. The score for ease of policy management is also two as detection rules can be easily changed but using the same inputs. The score for outsource solution is poor (one) as the system has been found massively dependant on using web services. The score for platform requirements are three as the system supports running on different platform and its hardware requirements are dependent on network volume traffic.

Item	Score
Distributed Management	2
Ease of configuration	2
Ease of policy management	2
Outsource Solutions	1
Platform Requirements	3

Table 6.1: Logistic Metrics

6.3 Design Metrics

6.3.1 Adjustable sensitivity

This parameter was defined by Fink, et al. (2002) as follow:

"Ability to change the sensitivity of the IDS to compensate for high false positive or false negative ratios."

By looking at the system, it has been found that the sensitivity of the system can be changed by modifying the fuzzy rules which define how to find suspicious IP addresses. Modifying the fuzzy rules using the same inputs specified in the previous chapter does not require a prior knowledge from the user, they are simple and human readable if then rules. In addition to the ability of modifying the fuzzy rules, the threshold value selected for the reasoning module output can play a role in adjusting the sensitivity of the system. Moreover, changing the black listed countries can have an impact in this area. The main disadvantage is the difficulty to consider new inputs in modifying the fuzzy rules to adjust the sensitivity. This requires adding membership functions to the fuzzifier which requires a prior knowledge of the fuzzy logic from the user.

6.3.2 Data storage

The system requires only to store fuzzy rules and black listed countries in a database, the size of this data does not exceed one Megabyte. This advantage is a result of using web services that get information about IP addresses, the alternative of using web services was to store information about IP addresses in a database and regularly update them. That database would include tables for IP geographic information, block listed IP addresses, and anonymous proxies. The size of such database would be around 1 Gigabytes.

6.3.3 Multi sensor support

In this context, sensor is defined as network sniffer that reads network packets coming from/to the system. The structure of the system supports having multi sensor, this is a result of using buffer that queues messages coming from network sniffing modules (sensors) and those messages are consumed by the IP information module. In addition, different sensors can be used with the system rather than the one implemented in the previous chapter, the sensor just needs to send a message containing an IP address.

6.3.4 Firewall Interaction

The reasoning module has been implemented in PHP which allows any person with prior knowledge of PHP to modify it as it is open source and not compiled files. Therefore, it is possible to modify it in a way that interacts with firewall based on the output it produces. For example, the reasoning module classifies an IP address as malicious so it necessary to add a firewall rules that blocks traffic from IP address.

6.3.5 Incident logging and notifications

As mentioned in the previous section, the system has been implemented using PHP. Thus, it can be modified in a way that suits an organization to notify and log incidents captured. For example, the system can be modified

to email the system administrator in case of capturing a traffic coming from a suspicious IP address.

6.3.6 Packet Loss

The proposed system uses TCPDUMP to capture network packets, this tool is very effective in monitoring tasks. However, it was reported by Antichi et al. (2014) in their research about monitoring high speed networks that software based on timestamping and capture such as TCPDUMP is not suitable to monitor traffic in scenarios where network speeds increases and this may lead to losing capturing some packets.

6.3.7 System throughput

The system throughput can be defined as how many packets the system can process per second. The system throughput depends on the environment it is running at. It has been found that the throughput of the system when running on the environment specified in table 6.2 was 10 packets/second.

CPU	Intel Core(TM) i7 CPU (2.1 GHz)
RAM	8 GB
Operating System	Windows 7

Table 6.2: Test environment for measuring system throughput`

6.3.8 Conclusion

Table 6.3 shows the score for each item in the design assessment. The score for adjustable sensitivity is two as it supports adjusting sensitivity through modifying the fuzzy rules but associated with some difficulties in some scenarios. The score of data storage is three as it does not require more than on Megabyte to store fuzzy rules and blacklisted countries in a database. The score for multi-sensor support is three as it does that with the ability to be communicated with different sensors than the one proposed

with the system. The score for both firewall interaction and incident logging/notification is also three at the system is an open source PHP code that can be easily modified. The score for packet loss is two as TCPDUMP can not perform well in high speed networks. The system throughput on the testing environment has achieved an acceptable rate so the score will be two for this item.

Item	Score
Adjustable sensitivity	2
Data Storage	3
Multi sensor support	3
Firewall Interaction	3
Incident Logging and notification	3
Packet loss	2
System Throughput	2

Table 6.3: Design Metrics

6.4 Performance Evaluation

6.4.1 Testing Data

The performance evaluation will involve two phases. The first one is evaluating whether the system is capable of detecting suspicious IP addresses, this will be achieved by building an IP list from different sources then go through each IP address in the list and apply it to the system and find out whether the system will classify as expected or not. The IP list will include suspicious addresses in different categories such as block listed, anonymous proxy or exit tor node in a predefined blacklisted countries list. In this experiment, the black listed country list will include China, Russia, and North Korea. The sources used in building this list are SANS (around

5,000 IP addresses), emerging threats (around 15,000 IP addresses), iblocklist around 61,000 IP addresses), and proxy nova (523 IP addresses for anonymous proxies). The second phase will test the system with some multi-stage attacks, this will be achieved by using trace files of captured traffic that involve multi stage attacks, the purpose of this phase is to prove that the system can detect multi stage attacks. The trace files were obtained from Computer Networks and Security research Group at Mugla Sitki Kocman University.

6.4.2 First Phase

As mentioned above, the aim of the first phase is to test whether the system is able to find suspicious IP addresses or not. The IP test list as shown in Table 6.4 includes different categories; Normal IP addresses in blacklisted countries, Normal IP addresses not in black listed countries, anonymous proxies in blacklisted countries, and block listed IP addresses.

Category	Number of IP addresses	Percentage
Normal IP addresses	10,000	10.99%
Anonymous proxy in a black listed country	523	0.57%
Block listed IP addresses	81,221	88.53%

Table 6.4: Different classes in the IP test list

A PHP script has been developed to perform the testing process. The scrip goes through the IP test list and applies each IP address to the IP info finder module which will then feeds the reasoning module with IP info. The output of the reasoning module is then compared with the expected result to update false positive, false negative, true positive, and true negative figures. Figure 6.2 shows the flow chart of the testing script.

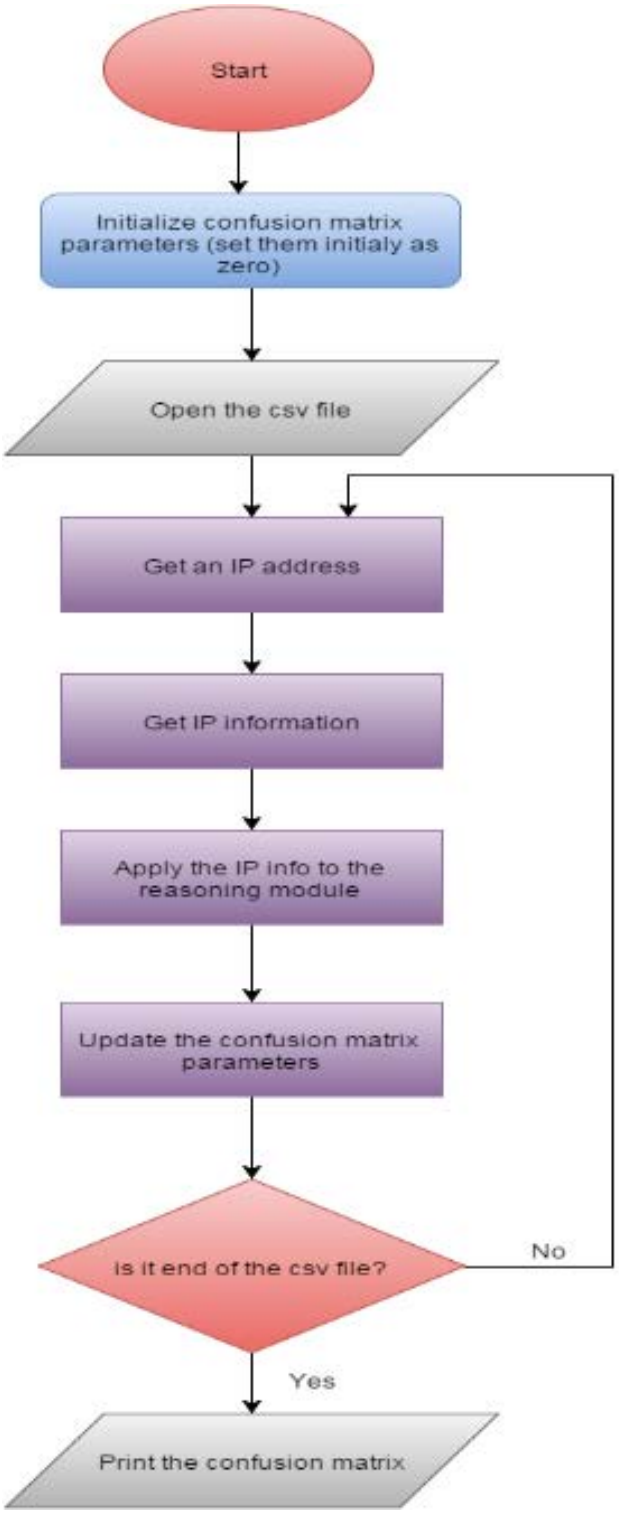

Figure 6.2: The flow chart of the testing proces

The results obtained from the first phase of the testing (see table 6.5) shows that all normal IP addresses are classified correctly (false positive is zero and true negative is one). On the other hand, the system managed to classify suspicious IP address with 0.994 true positive and 0.006 false negative. Most of IP addresses incorrectly classified are anonymous proxies located in black listed countries (357 IP addresses out of 523). This area can be improved by using another web service that is more sophisticated in finding anonymous IP addresses. One of those sophisticated web service is fraud lab (fraudlabs, 2015). By using the fraud lab web, the true positive went to 0.9984 as shown in table 6.6.

True Negative	False Positive
1	0
True Positive	**False negative**
.994	0.006

Table 6.5: The confusion metrics

True Negative	False Positive
1	0
True Positive	**False negative**
.9984	0.0016

Table 6.6: The confusion metrics after using the fraud lab web service to detect anonymous proxy

6.4.3 The Second Phase

6.4.3.1 SQL Attack Scenario

In this scenario, the attacker tried to perform a SQL injection in order to compromise a user on a web application. Table 6.7 shows IP participated in this scenario. If IP addresses are extracted from the trace files in the packets

order then apply them to the proposed system, the system will raise an alert from on the first packet. The system has found that 220.245.173.190 is a suspicious IP address (block listed IP address).

IP Addresses
220.245.173.190
12.25.187.58
12.25.187.61
12.25.187.255

Table 6.7: IP participated in the sql attack Scenario

6.4.3.2 UDP Scan Scenario

In this scenario, attackers performed a UDP scan using Nmap tool to get some information for further attack steps not included in the trace file. Table 6.8 shows IP participated in that attack. By applying the IPs extracted from the packets at the same order, the system raised an alert on the first packet, 24.6.173.220 has been found a suspicious IP address as it is block listed.

IP Address
24.6.173.220
74.207.244.221
74.207.244.221
24.6.173.220

Table 6.8: IP participated in the UDP scan Scenario

6.4.3.3 Exploiting Cross site Forgery Scenario

In this scenario, attackers exploited the cross site forgery vulnerability in a web application to change the password for a certain user. Table 6.9 shows IP participated in that scan. The system failed to raise an alert as none of IP address participated were classified as suspicious IP address.

IP Address
69.181.135.56
67.161.39.46

Table 6.9: IP participated in the cross site forgery scenario

6.4.3.4 Dictionary Attack Scenario

In this scenario, attackers performed a dictionary attack against FTP server. Table 6.10 shows IP participated in that scan. It has been found that the system raised an alert at the first packet indicating that 69.181.135.56 is a suspicious IP address (Block listed IP address).

IP Address
69.181.135.56
67.161.39.46

Table 6.10: IP participated in dictionary attack against FTP server

6.5 Conclusion

The evaluation has been divided into two phases. In the first phase, the metrics approach was followed. The metrics approach includes logistic,

design, and performance evaluation. In the logistic evaluation, it has been found that the score was medium (two) for supporting distributed management, ease of configuration, and ease of policy management. The system has a poor score in using outsource solutions while it achieved a high score in the platform requirements. By moving to the design metrics, it has been found that the system gets a high score in most of design criteria including data storage, multi sensor support, firewall interaction, incident logging and notification. However, it achieved a poor score in the system output. In addition, it is not performing well in high speed network and losses some packet due to using TCPDUMP in monitoring traffic. By looking at the performance evaluation, it has been found that the performance of the system in finding suspicious IP addresses after using the fraud lab web service to detect anonymous proxy.

The system was also tested with four real multi stage attack scenarios captured from a real network traffic. The system managed to capture three of them, while it failed to capture one of them as the IP addresses not classified as malicious. In the second phase, the proposed system was compared with solutions proposed by other researchers. It has been found that system does not require complex computation and memory resources compared to other solutions. On the other hand, it has been found that the main disadvantages is that it may not be able to capture an attack if IP addresses participated are not classified as suspicious while other solutions concerns about the attack logic not at identity of attacks sources. The proposed system is not a silver bullet for all multi stage attacks but it helps in reducing the occurrence of multi stage attacks. Introducing a system that based on event correlation and IP information (hybrid approach) will reduce the possibility of multi stage occurrence compared to using each approach individually.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

7.1 Overview

Many surveys and studies have shown the impact of cyber-attacks on organisations, governments, and individuals around the world. Although there are developments occurring in the computer security field, there are still cyber-attacks causing damage, because they are constantly being developed and evolved by attackers. This study has shown that improving intrusion detection methods is a vital element in enhancing the security of a system overall. This research investigated some industrial challenges in the intrusion detection area, identifying two main challenges; the first one is finding signature based intrusion detection systems such as SNORT lack the capability of detecting attacks with new signatures without human intervention. The other challenge is related to multi-stage attacks detection, and it has been found that signature-based is not efficient in this area. The contribution of this study has been in proposing methodologies that tackle the mentioned challenges.

7.2 Automatic Creation for SNORT Rules

As mentioned in the previous section, signature-based intrusion detection systems lack the capability of detecting attacks with new signatures. Part of the solution offered in this study has dealt with that issue through a multi-layer classification approach. In this approach, the first layer tries to classify the traffic either to normal or malicious. If the first layer fails to classify the traffic, the second layer (the reasoning module) will be triggered to classify the traffic. Both layers are based on data mining techniques. The signature holder will then be updated with the new attack signature. The first one was built using the J45 Decision Tree algorithm. The selection of this algorithm came after comparing results obtained by that algorithm with results obtained using the Naïve Bayes algorithm. The experiment conducted used the KDD'99 intrusion detection data set. The data were collected as a result of a setup of a fictitious military network with a number of target machines running various services. The data set consisted of 41 discrete and continuous attributes and had 22 attack classes and 1 normal class, where each instance in the data set was categorised as one class. A Correlation-based Feature Selection (CFS) algorithm was employed to evaluate the importance of each feature and provide a subset of features. The result of applying this algorithm in the data set was ten features. The selected features from the data set were used to train and evaluate both the Decision Tree and the Naïve Bayes. The results in the experiment showed that, although Naïve Bayes has a high TP, it is skewed by results of FP. On the other hand, the Decision Tree generates almost a consistent high TP and consistent low FP.

The reasoning module is based on a hybrid approach. Consisting of two modules; the first one is based on a neural network while the second one is based on fuzzy logic. The reasoning module will only classify network traffic as normal if both modules classify it as normal, while it will classify it as attack if either of them classifies the traffic as attack. A neural network has the advantage of the ability to work with incomplete and imprecise data.

This advantage can be employed in an IDS context for detecting attack patterns presented during the training phase but modified by an attacker in order to pass through the system. The flexibility of fuzzy logic can be employed in case of uncertain problems of intrusion detection and allows much greater complexity for IDS.

The benefit of using the hybrid approach is in increasing the intrusion detection rate; some of the attacks may not be detected by one of the modules but another one may be able to detect them. In other words, one module will overcome some of the other module's shortcomings in detecting malicious traffic. However, there is a chance of increasing the false positive rate for malicious traffic for the composite module, compared to each one individually. The data set used in the second layer is used to train the neural network and deduce the rules for the fuzzy logic module. It is then used to evaluate the reasoning module. The evaluation result showed that the true positive rate achieved was 0.966 with the neural network, 0.9995 with fuzzy logic, and 0.9997 when combining them together in one module. On the other hand, the false positive rate was .029 with the neural network, .022 with fuzzy logic, and increased to .048 with the composite module as expected.

Although good results were achieved in that area, some limitations have been found. One of those limitations is that both the neural network and Decision Tree modules may regularly require additional training to improve efficiency with detecting new patterns of attacks. The training process requires large set of data and consumes a lot of time. The fuzzy logic rules may also need to be updated in order to detect new patterns but it does not require as much time to get updated as the Decision Tree and neural network modules. The computer security context is always dynamic and changes dramatically. Thus, it is essential to update the modules used in the proposed approach when the true positive rate falls below a threshold value specified by the system administrator. In our case, all modules created in the experiments will definitely require an update if they are to be used with a

live system, as they were built using an old data set (KDD 1999). That data set was used to prove the concept of the approach. Part of this data set was used for training and creating fuzzy rules while the other part was used for evaluation to test whether the system was able to detect patterns not represented in the training process.

Another limitation that has been found is that fuzzy logic is relatively slow in processing the network traffic compared to the neural network and Decision Tree; this slowness is down to the size of fuzzy rules (1343 rules were used in the experiment). The inference engine needs to evaluate the rules with the fuzzy inputs provided, in order to produce an output. In addition to these limitations, the proposed approach will not help signature-based intrusion detection systems in improving multi-stage attack detection. Each step in multi-stage attacks looks legitimate and does not violate any rules with signature-based intrusion detection systems.

7.3 Multi-Stage Attack Prediction

The other industrial challenge that this study focused on was multi-stage attack detection. These attacks occur through multiple phases to get access to an organisation. Most of these attacks pass through three stages. In the first stage, attackers try to analyse available information about the target to find vulnerabilities and weaknesses that can be exploited. In the second stage, attackers exploit the weaknesses found in the first phase to inject malware into, or gain access to, the system. In addition, they try to get more details and conduct a deep analysis about the system to find data or resources they have an interest in. In the final phase attackers destroy the system or steal valuable information.

The study started looking at this industrial challenge by analysing four different multi-stage attack scenarios to understand the behaviour of multi-stage attacks and find any clue to predicting or detecting such kinds of attacks. In each scenario, the network traffic was analysed highlighting all

the steps that occurred and which were not considered by many security systems. The first scenario was about communication with a bad DNS server and how that can be exploited by an attacker to register machines to its bot army. The second scenario discussed the Shady Rat attack, which is a good example that shows how social engineering can be employed to target an organisation. The third scenario showed how header splitting can be employed by an attacker to target a network connected to a web host running a web application. The last scenario discussed how a vulnerable FTP service can be exploited to perform multi-stage attacks. An analysis of these four scenarios indicated that predicting such kinds of attacks may be achieved by carrying out a reputation check of IP addresses found in incoming and outgoing traffic.

The proposed approach in predicting multi-stage attacks is based on evaluating IP information. This approach was preferred over the event correlation- based approach, as the latter one requires having up-to-date multi-stage attack patterns (sequence), which is not easy to achieve in a very short time, as discovering new complex attacks normally takes some time. The Shady Rat Operation attack is a good example of that; it started in 2006 but was only discovered in 2011.

The proposed solution involves using three modules. The first one (network sniffing) is in charge of reading network traffic, and then extracting IP addresses from the packets. This module was built by using the TCPDUMP tool. The TCPDUMP was preferred over other sniffing tools, as it is available on many operating systems. In addition, it can be easily integrated into the proposed solution, as it is command line-based.

The second module (IP information finder) was created to gather information about IP addresses extracted from the first module. There were two options to obtain the IP information. The first option was by storing IP information in a database, then checking the IP address against it to find any associated information. The second option was to obtain the IP information

through one of the available web services (API). The main disadvantage of this first solution was the need to be regularly updated. On the other hand, there are many web services regularly updated and, therefore, it was decided to go with the second option. The information obtained by the second module involved finding the IP geographic location, checking whether the IP was on a block list, whether the IP was an anonymous proxy, or a TOR exit node, and checking the IP rating in DNSBL. The last module (reasoning) was responsible for deciding whether the IP addresses may be a source of malicious traffic or not, based on information passed from the second module. The outputs of the second module need to be pre-processed before processing them.

The reasoning module was implemented using fuzzy logic. The reason for choosing fuzzy logic, rather than any of the machine learning algorithms, was the nature of the problem; it can be simply solved using "if… then" rules. In addition, it requires less time and effort to adapt to changes in the reasoning logic compared to machine learning algorithms. Machine learning algorithms require large sets of training data to get accurate results and that consumes a lot of time in the training phase. The fuzzy logic was initially tested with four rules. The first rule stated that the IP will be considered as a malicious one if the IP address is found in a block list. Having an IP in a block list implies that the IP address was reported as being used in malicious traffic. The second and third were similar, with each rule involving the checking of two parameters. One of them was finding whether the IP was based in one of the black listed countries or not. Being in a black listed country does not mean that the traffic is malicious, as there may be legal traffic from those countries. The second parameter checked whether an anonymous proxy or exit tor node was used or not. Anonymous proxies and exit tor nodes are used in a way that enables users to protect their access to the web anonymously. Attackers normally do not necessarily need to be based in the listed countries, they forward their traffic through a proxy or tor located in one of those countries. Thus, receiving traffic from

anonymous proxies or tor exit nodes located in those countries raises an alert of potential malicious traffic. The last rule was to find out the average IP rating (the host reputation). The IP address is treated as malicious if the average rating is low.

The evaluation process of the proposed approach was carried out using the metrics based approach which looked at the evaluated approach from the different perspectives of logistic, design, and performance. The last metric used in this approach was the confusion metric (performance metric), which finds how well the system does its job (detecting multi-stage attacks) in the form of true positive, true negative, false positive, and false negative. The logistic metrics evaluates the system in terms of maintainability, manageability, and dependency. It was found that the proposed approach got a medium score from the logistics perspective. On the other hand, it had a high score when looking from the design perspective that measured how well the approach performed in terms of resources consumption and speed. Regarding the performance, it was first measured using a list of 91,744 IP addresses, including different categories (10.99% Normal, 0.57% anonymous proxy in a black listed country, 88.53% block listed IP addresses) to ensure that the approach was capable of distinguishing between malicious and normal IP addresses. It was found that the system achieved a good performance with zero false positive and a high true positive rate (0.9984). However, it was found in the second stage of the performance evaluation that it failed to detect multi-stage attack scenarios, if the IP addresses participating in the traffic were not classified as malicious IP addresses. That stage involved testing the approach with four different multi-stage attack scenarios.

When comparing the proposed solution with other solutions based on event correlation, it has an advantage over them by not being dependent on receiving alerts from IDS. In addition, it does not require a complex computation, or memory resources, compared to them. Furthermore, the previous solutions required an update with sequences of new attacks, while

the proposed solution focuses on the identity. However, this may represent an issue, if an attack comes from an IP address not classified yet as suspicious. Moreover, the throughput of the proposed system is relatively low compared to other solutions, due to using web services that take some time to get IP information.

7.4 Future Work

It is planned to overcome some of the limitations found in the proposed approaches during the experiments. One of those limitations is the slowness found with the fuzzy logic module created in the system that automates the creation of automatic rules. The work to be carried out in this area will involve optimising a number of rules used with the fuzzy logic modules. In addition, it is intended to re-create both the Neural Network and fuzzy logic modules using all classes, and then compare the results with results obtained in this study. Moreover, different data mining techniques will be tried with fuzzy logic and Neural.

Moving to the multi-stage detection area, it is planned to improve the IP information module by using alternative web services, in case the web service provided by Neutrino fails. It can be investigated whether there is more IP information that can be used in identifying malicious traffic. The proposed approach in this research can also be combined with an event correlation-based approach to gain advantage of both approaches by looking at both identities and traffic content.

7.5 Publications Related to this Thesis

1. Almutairi, A. Parish, D. Flint, J. “Predicting Multi-Stage Atta Based on IP Information ” in Proc. of Internet Technology and Secured Transactions conference, vol.8, no.1, pp. 42-50, 2015.
2. Almutairi, A. and Parish, D. “Improving Intrusion Detection by the Automated Generation of Detection Rules” in International Journal of Intelligent Computing Research, vol.5, no.1, pp. 495-502, 2014.
3. Almutairi, A. and Parish, D. “Using classification techniques for creation of predictive intrusion detection model” in Proc. of Internet Technology and Secured Transactions conference, pp. 223-228, 2014.
4. Almutairi, A. And Parish, D. “Survey of High Interaction Honeypot Tools: Merits and Shortcomings” in Proc. of PGNET, vol.8, pp. 42-50, 2012.
5. Almutairi, A., Flint, J. and Parish, D., 2015. Predicting multi-stage attacks based on hybrid approach. International Journal for Information Security Research, 5 (3), pp. 582 - 590.

References

1. Abbes, T., Bouhoula, A., and Rusinowitch, M. (2004) Protocol analysis in intrusion detection using decision tree ITCC 2004. International Conference 1(April 2004), pp. 404-408
2. Abraham, T., IDDM: Intrusion detection using data mining techniques. 2001.
3. Alberto, P., Sala, A. and Olivares, M. “Fuzzy Logic Controllers. Methodology. Advantages and Drawbacks” [Online] Available from: http://www.softcomputing.es/estylf08/es/2000-X%20Congreso/01%20SESION%20INAUGURAL.pdf [Accessed: 19th April 2015]
4. Albin, E. (2011) A COMPARATIVE ANALYSIS OF THE SNORT AND SURICATA INTRUSION-DETECTION SYSTEMS, NAVAL POSTGRADUATE SCHOOL.
5. Allen, W. H. 2007. Mixing wheat with the chaff: Creating useful test data for IDS evaluation. IEEE Security & Privacy, 65-67.
6. Al-Mamory, S. and Zhang, H. (2008) New data mining technique to enhance IDS alarms quality. Journal in Computer Virology , 6(1), pp 43-55
7. Alserhani, F. et al. (2010): "Multi-Tier Evaluation of Network Intrusion Detection Systems" Journal for Information Assurance and Security (JIAS), 5 (4): 301-310.
8. Altwaijry, H., Bayesian based intrusion detection system, in IAENG Transactions on Engineering Technologies2013, Springer. p. 29-44.
9. Amiri, F., et al., Mutual information-based feature selection for intrusion detection systems. Journal of Network and Computer Applications, 2011. 34(4): p. 1184-1199

10. Amor, N., Benferhat, S. and Elouedi, Z. (2004) Proceeding SAC'04 Proceedings of the 2004 ACM symposium on Applied computing, PP. 420-424.
11. Amro, S. et al. 2012. Evolutionary Computation in Computer Security and Forensics: An Overview. Computational Intelligence for Privacy and Security, 25-34.
12. Anthony, C. (2014) ARTIFICIAL NEURAL NETWORKS IN PROTEIN SECONDARY STRUCTURE PREDICTION: A CRITICAL REVIEW OF PRESENT AND FUTURE APPLICATIONS, Stanford University
13. Anthony, M. and Bartlett, P. “Neural Network Learning: Theoretical Foundations” 2009, Campridge University.
14. Antichi, G. et al. Enabling open-source high speed network monitoring. In Proceedings of NetFPGA IEEE 2012 Network Operations and Management Symposium, pp.1029-1035, April 2012
15. Axelsson, S. The base-rate fallacy and its implications for the difficulty of intrusion detection. in Proceedings of the 6th ACM Conference on Computer and Communications Security. 1999. ACM.
16. Bace, R. and Mell, P. (2001) Intrusion Detection System. National Institute of Standards and Technology, Available from http://www.cryptome.org/
17. Baecher, P. et al. The nepenthes platform: An efficient approach to collect malware. Recent Advances in Intrusion Detection, 2006. Springer, 165-184.
18. Barbara, D., et al., ADAM: a testbed for exploring the use of data mining in intrusion detection. ACM Sigmod Record, 2001. 30(4): p. 15-24.
19. Batal, I. (2014) Dimensionality Reduction techniques, University of Pittsburgh. Available From http://people.cs.pitt.edu/~iyad/

20. Bellard, F. Qemu, a fast and portable dynamic translator. 2005. USENIX.
21. Bhargava, N., et al., Decision Tree Analysis on J48 Algorithm for Data Mining. International Journal, 2013. 3(6).
22. Bidgoli, B., Analoui, M., Rezvani, M. ,and Shahhosein, H. (2008) Performance Evaluation of Decision Tree for Intrusion Detection Using Reduced Feature Spaces. Iran University of Science and Technology
23. Borah, S. and A. Chakraborty, Towards the Development of an Efficient Intrusion Detection System. International Journal of Computer Applications, 2014. 90.
24. Bouzida, Y. and Cuppens, F. Neural networks vs. decision trees for intrusion detection.
25. Bradely, T. Introduction to Intrusion Detection Systems (IDS). About Technology, Available from http://netsecurity.about.com/
26. Bridges, S. and Vaughn, R. (2000). FUZZY DATA MINING AND GENETIC ALGORITHMS APPLIED TO INTRUSION DETECTION. Department of Computer Science Mississippi State University
27. Castro, L. and Zuben, F. (2002) Learning and Optimization Using the Clonal Selection Principle. IEEE Transactions on Evolutionary Computation, 6 (3), pp. 239–251
28. Chae, H. and Choi, S. (2014) Feature Selection for efficient Intrusion Detection using Attribute Ratio. INTERNATIONAL JOURNAL OF COMPUTERS AND COMMUNICATIONS, 8(2014)
29. Chang, R. et al., “Intrusion detection by backpropagation neural networks with sample-query and attribute-query,” International Journal of Computational Intelligence Research, 2007, vol.3, n.1, pp. 6-10.

30. Chang, R.-I., et al., Intrusion detection by backpropagation neural networks with sample-query and attribute-query. International Journal of Computational Intelligence Research, 2007. 3(1): p. 6-10.
31. Chitrey, A. "Apprehensive Study of Social Engineering Based Attacks in India to Develop a Conceptual Model" IJINS ,2012. vol.1, n.4
32. Chittur, A. (2001) Model generation for an intrusion detection system via genetic algorithms [Online] Available from http://www.hacktory.cs.columbia.edu [Accessed 15th September 2014]
33. Chou, T. (2007), Correlation-Based Feature Selection for Intrusion Detection Design. Military Communications Conference, 2007. MILCOM 2007. IEEE, pp. 1-7
34. Clark, D. "The Problem isn't Attribution; It's Multi-Stage Attacks" the Re-Architecting the Internet Workshop, 2010, Article No.11
35. Corchado, E. and Å. Herrero, Neural visualization of network traffic data for intrusion detection. Applied Soft Computing, 2011. 11(2): p. 2042-2056.
36. Corey, J. 2003. Local honeypot identification. Fake Phrack Magazine http://www.ouah.org/p62-0x07.txt.
37. Crosbie, M., and Spafford, E. (1995) Applying Genetic Programming to Intrusion Detection, Proceedings of the AAAI Fall Symposium.
38. Cunningham, R. and Lippmann, R. (2000) Improving Intrusion Detection performance using Keyword selection and Neural Networks. MIT Lincoln University Available from http://www.ll.mit.edu/IST/pubs.html
39. Davis, J.J. and A.J. Clark, Data preprocessing for anomaly based network intrusion detection: A review. computers & security, 2011. 30(6): p. 353-375.

40. Derban, G. and Moldovan, G. (2006) A comparison of clustering techniques in aspect mining. Studia University, (51), pp. 69-78.
41. Einwechter, N. (2002) An Introduction To Distributed Intrusion Detection Systems [Online] Available from: http://www.symantec.com/connect/articles/introduction-distributed-intrusion-detection-systems [Accessed: 22nd July 2015]
42. Ertoz, L., et al. 2004. Minds-minnesota intrusion detection system. Next Generation Data Mining, 199-218.
43. Escamilla, T. (1998) Intrusion Detection: Network Security Beyond the Firewall. New York: John Wiley and Sons.
44. Ethereal. 2012. Ethereal: A Network Protocol Analyser [Online]. Available: http://www.ethereal.com [Accessed 1st March 2012].
45. Fink, A. "A Metrics-Based Approach to Intrusion Detection System Evaluation for Distributed Real-Time Systems" Information TransferTechnology Group, 2002, Code B35, Naval Surface Warfare Center, Dahlgren Division
46. Gabra, H., Baha-Eldin, A., and Korshi, H. (2014) Classifications of IDS Alertys with Data Mining Techniques. International Journal of Electronic Commerce Studies, 5(1) pp. 1-2
47. Gadelrab, M. et al. Defining categories to select representative attack test-cases. 2007. ACM, 40-42.
48. Garcia-Teodoro, P. et al. 2009. Anomaly-based network intrusion detection: Techniques, systems and challenges. computers & security, 28, 18-28.
49. GCHQ and Cert-UK (2015) Common Cyber Attacks: Reducing The Impact. [Paper]. United Kingdom: GCHQ and Cert-UK
50. Ghosh, A. and Schwartzbard, A. (2000) A study using Neural Networks for anomaly detection and misuse detection. Reliable Software Technologies, Available from http://www.docshow.net/ids/

51. Grossman, L. Data Mining (1997) Challenges and Opportunities for Data Mining During the Next Decade. [Online] Available from: http://www.lac.uic.edu [Accessed 19th August 2014]
52. GUPTA, P. et al. 2012. Securing WMN using Honeypot Technique. International Journal on Computer Science and Engineering, 4, 235-238.
53. Hall, B. 2011 Countering Web Injection Attacks: A Proof of Concept, MSc thesis, University of Manchester UK.
54. Hawrylkiw, D. Network Intrusion and use of automated responses. [Online] Available from: http://www.sans.org/resources/idfaq/auto_res.php, [Accessed 20th August 2015]
55. Hall, M. (1999) Correlation-based Feature Selection for Machine Learning. University of Waikato.
56. Hall, M., et al., The WEKA Data Mining Software: An Update. SIGKDD Explorations, 2009. 11(1).
57. Hall, M.A., Correlation-based feature selection for machine learning, 1999, The University of Waikato. (Hall, 1999)
58. Hassan, M. (2011) Difference Between Firewall and Intrusion Detection System, [Online] Available from http://www.scribd.com/ [Accessed 18th August 2014]
59. Hes, R., Engineering, V. U. O. W. S. O. & Science, C. 2009. The Capture-HPC client architecture, School of Engineering and Computer Science, Victoria University of Wellington.
60. Ho, C. Lin, Y. Lai, Y. Chen, I. Wang and Tai, W. (2012) False Positives and Negatives from Real Traffic with Intrusion Detection/Prevention Systems. International Conference on Advancements in Information Technology, pp. 1-4
61. Huang, P., Yang, C. & Ahn, T. Design and implementation of a distributed early warning system combined with intrusion detection system and honeypot. Proceedings of the 2009 International

Conference on Hybrid Information Technology ICHIT '09 2009. ACM, 232-238.

62. I-Blocklist, List of blocklisted IP addresses, [Online] Available from: https://www.iblocklist.com/ [Accessed: 24th July 2015]
63. J. Muila, A Novel Intrusion Detection System (IDS) Architecture, 2010
64. Ji, Yu, Zhang, (2011). A novel Naive Bayes model: Packaged Hidden Naive Bayes. Information Technology and Artificial Intelligence Conference, 2(Aug 2011), pp. 484 -487
65. Jiang, X. & Wang, X. Out-of-the-box monitoring of VM-based high-interaction honeypots. Proceedings of the 10th international conference on Recent advances in intrusion detection RAID'07 2007 Gold Goast, Australia. Springer-Verlag, 198-218.
66. Jyothsna, V., Prasad, V. and Prasad, K. (2011) A Review of Anomaly based Intrusion Detection Systems. International Journal of Computer Applications, 28(7)
67. K. Labib and R. Vemuri (2002) NSOM: a real-time network-based intrusion detection system using self-organizing maps, Networks and Security
68. Kachal, C. and Shevade, K. (2012) Comparison of Different Intrusion Detection and Prevention Systems. International Journal of Emerging Technology and Advanced Engineering, 2(12)
69. Kailashiya, D. and Jain, R. (2012) Improve Intrusion Detection Using Decision Tree with Sampling. International Journal of Computer Science and Telecommunications, 3(3), pp. 1209
70. Kamepalli, S. and Mothukuri, R. (2014) Implementation of Clustering-Based Feature Subset Selection Algorithm for High Dimentional Data. International Journal of Emerging Trends & Technology in Computer Science, 3(3)
71. Kang, D.-K., D. Fuller, and V. Honavar. Learning classifiers for misuse and anomaly detection using a bag of system calls

representation. in Information Assurance Workshop, 2005. IAW'05. Proceedings from the Sixth Annual IEEE SMC. 2005. IEEE.

72. Kayacik, H. G., Zincir-Heywood, A. N. & Heywood, M. I. 2012. Intrusion Detection Systems. Signal Processing, 1pp.
73. KDD'99. KDD Cup 1999 Data. 1999 [cited 2014; Available from: https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.
74. KFSENSOR. 2012. KFSensor: Advanced Windows Honeypot System [Online]. Ket Focus. Available: http://www.keyfocus.net/kfsensor/ [Accessed 15th March 2012].
75. Kim, G., S. Lee, and S. Kim, A novel hybrid intrusion detection method integrating anomaly detection with misuse detection. Expert Systems with Applications, 2014. 41(4): p. 1690-1700.
76. Kukiełka, P. and Kotulski, Z. (2010) "Adaptation of the neural network-based IDS to new attacks detection", Available from: http://arxiv.org/abs/1009.2406 (Access Date: 17 Oct, 2014).
77. Kukiełka1, K. and Kotulski, Z. (2010) . Analysis of neural networks usage for detection of a new attack in IDS Annales UMCS Informatica AI X, 1 (2010) 51-59
78. Kumar and Guyal (2004) GA-NIDS: A Genetic Algorithm based Network Intrusion Detection System, Northwestern University
79. Kuwatly, I. et al. A dynamic honeypot design for intrusion detection. IEEE/ACS International Conference on Pervasive Services, 2004. ICPS 2004, 2004. IEEE, 95-104.
80. LAZAREVIC, A. et al. comparative study of anomaly detection schemes in network intrusion detection. 2003. SIAM, 25-36.
81. Lee, W. and S.J. Stolfo, A framework for constructing features and models for intrusion detection systems. ACM transactions on Information and system security (TiSSEC), 2000. 3(4): p. 227-261.
82. Liao, H.-J., et al., Intrusion detection system: A comprehensive review. Journal of Network and Computer Applications, 2013. 36(1): p. 16-24.

83. Liaoa, H., Lina, C. Lina, Y. and Tunga, K. (2012) Intrusion detection system: A comprehensive review. Journal of Network and Computer Applications.
84. Lindfors, A. and Peuhkuri, M. “Vulnerabilities of FTP protocol, FTP servers and clients” 1999, Department of Electical Engineering, Helsinki University of technology
85. Lippmann, R. et al. Evaluating intrusion detection systems: The 1998 DARPA off-line intrusion detection evaluation. 2000. IEEE, 12-26 vol. 2.
86. Lo, O., Graves, J. & Buchanan, W. J. Towards a framework for the generation of enhanced attack/background network traffic for evaluation of network-based intrusion detection systems. 2010. Academic Conferences Limited, 190.
87. Mabu, S., et al., An intrusion-detection model based on fuzzy class-association-rule mining using genetic network programming. Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, 2011. 41(1): p. 130-139.
88. Mahoney, M. & Chan, P. An analysis of the 1999 DARPA/Lincoln Laboratory evaluation data for network anomaly detection. 2003. Springer, 220-237.
89. Majeed, P. and Kumar, S. (2014) Genetic Algorithms in Intrusion Detection Systems: A Survey. International Journal of Innovation and Applied Studies, 5(Mar 2014), pp. 233-240
90. Makkithaya, M., Reddy, N., and Acharya, U. (2008) Improved C-Fuzzy Decision Tree for Intrusion Detection. World Academy of Science, Engineering and Technology, 2(6)
91. Markey, J. (2011) Using Decision Tree Analysis for Intrusion Detection: A How-To Guide. Sans Institute, Available from http://www.sans.org/reading-room

92. Mathworks – What Is Mamdani-Type Fuzzy Inference? [Online] Available from: http://uk.mathworks.com/help/fuzzy/what-is-mamdani-type-fuzzy-inference.html. [Accessed: 26th April 2015]
93. Mcafee. Operation Shady Rat – What It Really Means, and What You Can Learn From It? [Online] Available from: http://www.symantec.com/connect/blogs/truth-behind-shady-rat/ . [Accessed: 25th Feb 2015]
94. MCHUGH, J. 2000. Testing intrusion detection systems: A critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory. ACM transactions on Information and system Security, 3, 262-294.
95. Mehra, P. (2012) A brief study and comparison of SNORT and BroOpen Source Network Intrusion Detection Systems. International Journal of Advanced Research in Computer and Communication Engineering, 1(6).
96. Meijerink, M. & Spellen, J. 2006. Intrusion Detection System honeypots. Master Program System and Network Administration, University of Amsterdam.
97. Mell, P., et al. 2003. An overview of issues in testing intrusion detection systems.
98. Mitnick, K. (2002) How to hack people", BBC News Online
99. Mokube, I. & Adams, M. Honeypots: concepts, approaches, and challenges. Proceedings of the 45th annual southeast regional conference ACM-SE 45, 2007 Winston-Salem, North Carolina. ACM, 321-326.
100. Muila, J. A Novel Intrusion Detection System (IDS) Architecture, 2010
101. Mujahid, T. and Mathew, T. (2014) A Genetic Algorithm Analysis towards Optimization solutions, International Journal of Digital Information and Wireless Communications 4(1), pp. 124-142

102. Naudts, J. (2004) Improvement of Weka, a datamining tool. Universiteit Gent
103. Nazer, G. M. & Selvakumar, A. A. L. 2012. Intelligent Data Mining Techniques for Intrusion Detection Models on Network. European Journal of Scientific Research, 71, 36-45.
104. NETSEC. 2012. Specter [Online]. Netsec. Available: http://www.specter.com/default50.htm [Accessed 15th March 2012].
105. Nikolova, E. & Jecheva, V. 2011. Evaluations of the effectiveness of anomaly based intrusion detection systems based on an adaptive knn algorithm.
106. Ourston, D. et al. Applications of hidden markov models to detecting multistage network attacks. In Proceedings of the 36th Hawaii International Conference on Systems Sciences, Los Alamitos, CA, USA, 2003.
107. Paliwal, S. and Gupta, R. Denial-of-Service, Probing & Remote to User (R2L) Attack Detection using Genetic Algorithm. International Journal of Computer Applications 60(19), pp. 57-62.
108. Panda, M. and Patra, M. (2007) Network intrusion deteciton using naïve Bayes. International Journal of Computer Science and Network Security, 7(12), pp. 258-263.
109. Attacking Fhttps://pentestlab.wordpress.com/2012/03/01/attacking-the-ftp-service/ [Accessed 7th May 2015]
110. Phung, M. (2000) Intrusion Detection FAQ: Data Mining in Intrusion Detection, SANS Institue, Available from http://www.sans.org/
111. Ponemon Institue(2014) Global Report on the Cost of Cyber Crime. [Benchmark study]. United States: Ponemon Institue
112. Portokalidis, G., Slowinska, A. & BOS, H. 2006. Argos: an emulator for fingerprinting zero-day attacks for advertised honeypots with automatic signature generation. ACM SIGOPS Operating Systems Review, 40, 15-27.

113. Potts, S. and Kopack M. 2003, Web Service. United States: Sams Publishing
114. Prediction Works , Data Mining Glossary, Available from http://www.predictionworks.com/glossary
115. Project, T. H. 2012. The Honeynet Project [Online]. Available: http://www.honeynet.org [Accessed 1st March 2012].
116. Provos, N. Honeyd-a virtual honeypot daemon. 10th DFNCERT Workshop, 2003.
117. Proxynova, List of anonymous proxy, [Online] Available from: https://www.proxynova.com/ [Accessed: 25th July 2015]
118. Pulo, K. “Fuzzy Logic vs Machine Learning” , [Online] Available from: http://www.kev.pulo.com.au/ai/fuzzyml_report/ [Accessed: 20th April 2015]
119. Rajasekaran, S. and Pai, G. “Neural Networks, Fuzzy |Logic and Genetic Algorithm: Synthesis and Applications” 2003, PHI Learning Pvt. Ltd.
120. Rani, S. and Singh, V. (2014) An Open Source Network Security Tool for Intrusion Detection in Campus Network Environment, International Journal of Computer Technology and Electronics Engineering, 2(1)
121. Rawat, R. and A. Jain, Review: Boosting Classifiers For Intrusion Detection. nternational Journal of Scientific & Engineering Research, 2013. 4(7): p. 1-5.
122. Rawlings, N. (2013) Anonymous Hackers Plead Guilty to PayPal Cyber Attack. [online] United : Time. Available from http://techland.time.com/2013/12/09/anonymous-hackers-plead-guilty-to-paypal-cyber-attack/ [Accessed 2nd September 2015].
123. Reddy, Kesavulu., Reddy, V., and Rajulu, P. (2011) A Study of Intrusion Detection in Data Mining. the World Congress on Engineering, 3(July 2011)

124. Reddy, P. (2011) Data Mining Machine Learning Techniques – A Study on Abnormal Anomaly Detection System. International Journal of Computer Science and Telecommunications, 2(6), pp. 8-14
125. Roesch, M. SNORT: Lightweight Intrusion Detection for Networks. in LISA. 1999.
126. Rokach, L. and Mimon, O. (2014) Data Mining with Decision Trees. 2nd Edition, Massachusetts: World Scientific
127. Rossey, L. et al. Lariat: Lincoln adaptable real-time information assurance testbed. 2002. Ieee, 6-2671-2676, 6-2678-6-2682 vol. 6.
128. Ruggieri, S. Efficient C4.5. IEEE Transactions on Knowledge and Data Engineering. Vol. 14, Issue 2, March-April 2002, 438-444.
129. Saber, M., Bouchentouf, T. & Benazzi, A. Generation of attack scenarios by modeling algorithms for evaluating IDS. 2011. IEEE, 1-5.
130. Sadasivam, K., Samudrala, B. & Yang, T. A. 2005. Design of network security projects using honeypots. Journal of Computing Sciences in Colleges, 20, 282-293.
131. Sagane, A. and Dhande, S. (2014) Malicious Code Detection Using Naïve Bayes Classifier. International Journal of Application or Innovation in Engineering & Management, 3(4)
132. Saini, H., Mishra, B. K., Pratihari, H. & Panda, T. 2011. Extended Honeypot Framework to Detect old/new cyber attacks. International Journal of Engineering Science (IJEST), 3, 2421-2426.
133. Salah, K., Al-Khiaty, M. , Ahmed, R. , Mahdi, A. (2011) Performance Evaluation of SNORT under Windows 7 and Windows Server 2008.Journal of Universal Computer Science, 17(11), pp. 1605-1622
134. Sans Block List [Online] Available from: https://isc.sans.edu//block.txt. [Accessed: 26th April 2015]
135. SANS Institute (2001) Understanding Intrusion Detection Systems.

136. SANS Institute (2007) Social Engineering: A Means To Violate A Computer System
137. SANS Isstitue (2001) Intrusion Detection Systems: Definition, Need and Challenges.
138. Scarfone, K. and Mell, P. (2007) Guide to Intrusion Detection and Prevention Systems (IDPS). National Institute of Standards and Technology.
139. Shanmugam, B. “Improved Intrusion Detection System Using Fuzzy Logic for Detecting Anamoly and Misuse Type of Attacks” in Proceedings of the Conference of Soft Computing and Pattern Recognition. 2009, pp.212-217.
140. Sharma, N. & Sran, S. S. 2011. Detection of threats in Honeynet using Honeywall. International Journal on Computer Science and Engineering (IJCSE), 3, 3332-3336.
141. Singh, A. N. & Joshi, R. A honeypot system for efficient capture and analysis of network attack traffic. International Conference on Signal Processing, Communication, Computing and Networking Technologies (ICSCCN), 2011, 2011. IEEE, 514-519.
142. Singh, S. (2010) Data Clustering Using K-Mean Algorithm for Network Intusion Detection. Lovely Proffessional University.
143. Sommers, J., Yegneswaran, V. & Barford, P. Recent 2005. Toward comprehensive traffic generation for online ids evaluation. University of Wisconsin, Tech. Rep.
144. Sommers, J., Yegneswaran, V. & Barford, P. Recent A framework for malicious workload generation. 2004. ACM, 82-87.
145. Sommers, J., Yegneswaran, V. & Barford, P. Recent advances in network intrusion detection system tuning. 2006. Ieee, 1490-1495.
146. Spadaro, A. 2013 Event correlation for detecting advanced multi-stage cyber-attacks, MSc thesis, Delft University of Technology Netherlands.

147. Spitzner, L. 2000. Know your enemy: A forensic analysis. http://www.rootprompt.org/article.php3?article=520.
148. Spitzner, L. 2003. Honeypots: simple, cost-effective detection. SecurityFocus InFocus Article, http://www.securityfocus.com/infocus/1690.
149. Statistia(2015) Costs of external consequences of cyber attacks on businesses 2015. [Survey] United States: Statistia
150. Sterling, T. and Escritt, T. (2015) Dutch government website outage caused by cyber attack. [online] London: Reuters. Available from http://www.reuters.com/article/2015/02/11/us-netherlands-government-websites-idUSKBN0LF0N320150211#2vGIVmOh5yBuYCdH.97 [Accessed 29th August 2015].
151. Swanson, I. Malware, Viruses and Log Visualisation. 2008. 54.
152. Tal Global. (2011) Operation Shady Rat – What It Really Means, and What You Can Learn From It? [Online] Available from: http://talglobal.com/operation-shady-rat-what-it-really-means-and-what-you-can-learn-from-it/ . [Accessed: 19th Feb 2015]
153. Tan, P., Steinbach, M., and Kumar, V. (2006) Introduction to Data Mining. Boston: Pearson.
154. Taruna, S. and Hiranwal, S. (2013) Enhanced Naïve Bayes Algorithm for Intrusion Detection in Data Mining. International Journal of Computer Science and Information Technologies, 4(6) , pp. 960-962
155. TCPDUMP. 2012. TCPDUMP [Online]. TCPDUMP. Available: http://www.TCPDUMP.org [Accessed 16th March 2012].
156. Templeton S. and Levit. K. A requires/provides model for computer attacks. In Proc. of New Security Paradigms Workshop, pages 31 – 38. September 2000.

157. TP Group, Network Trace Files [Online] Available from: http://www.tp.org/jay/nwanalysis/traces/General%20Trace%20Files/ [Accessed: 24th Feb 2015].
158. Vora, P., Oza, B. (2013) A Survey on K-mean Clustering and Particle Swarm Optimization. International Journal of Science and Modern Engineering. 1(3)
159. Wahbeh, A., Al-Radaideh, Q., Al-Kabi, M., and Shawakfa, E. (2011) International Journal of Advanced Computer Science and Applications, Special Issue on Artificial Intelligence, pp. 18-26
160. Wang, G., et al., A new approach to intrusion detection using Artificial Neural Networks and fuzzy clustering. Expert Systems with Applications, 2010. 37(9): p. 6225-6232.
161. Weber, D. (1998) A Taxonomy of Computer Intrusions. Master's thesis, Massachusetts Institute of Technology
162. WEI, D.-z., Q.-g. WANG, and L.-n. LIN, Design and Application of a SNORT System Based on Data Mining. Journal of Jimei University (Natural Science), 2011. 5: p. 016.
163. Wei, M., Xia, L., and Su, J. (2011) Research on the Application of Improved K-Means in Intrusion Detection. Communications in Computer and Information Science, 243(2011), pp. 673-678
164. Werlinger, R. The challenges of using an intrusion detection system: is it worth the effort?. In Proc. of the 4th symposium on Usable privacy and security, pages 107 – 118. 2008.
165. Werlinger, R., Hawkey, K., Muldner, K., Jaferian, P., and Beznosov, K. (2008) The Challenges of Using an Intrusion Detection System: Is It Worth the Effort? University of British Columbia, pp.1-2
166. Williams, R.(2014) Cyber crime costs global economy $445 bn annually. [online] London: Telegraph. Available from http://www.telegraph.co.uk/technology/internet-security/10886640 [Accessed 18th August 2015].

167. Zanero, S. & Savaresi, S. M. Unsupervised learning techniques for an intrusion detection system. 2004. ACM, 412-419.

Appendix A: Project plan

A.1 Project Management Methodology

The project management methodology used in this project is a modified approach of Method123 project management methodology, which is a Prince2 based methodology. Method123 includes some activities which are used with medium and large sized projects. Some of those activities not applicable to this research project. Thus, Method123 has been tailored to suit the project's size and requirements.

A.2 Project Schedule

Table A.1 shows the break down structure of the project, detailing its phases, and activities while figure 1.3 shows the Gantt chart. This illustrates the project schedule, showing that the project requires n working days for completion.

	Activity	Start	End
Literature survey	Investigation about honeypot	11/01/2011	09/30/12
	Investigation about IDS	11/01/2011	12/31/11
	Investigation about data mining techniques	01/01/2012	03/31/12
	Preparation for a conference	04/01/2012	07/31/12
	Writing the first year report	08/01/2012	08/31/12
Experiment 1	Requirement Analysis	09/01/2012	09/30/12
	Design	10/01/2012	09/30/13
	Implementation	10/01/2012	11/15/12
	Evaluation	11/16/12	12/31/12
	Preparation for a conference	01/01/2013	05/31/13
	Writing the second year report	06/01/2013	07/31/13
Experiment 1	Requirement Analysis	08/01/2013	08/31/13
	Design	09/01/2013	09/30/13
	Implementation	10/01/2013	09/30/14
	Evaluation	10/01/2013	11/15/13
	Preparation for a conference	11/16/13	12/31/13
	Writing the second year report	01/01/2014	05/31/14
Writing UP	Chapter 2	06/01/2014	07/31/14
	Chapter 3	08/01/2014	08/31/14
	Chapter 4	09/01/2014	09/30/14
	Chapter 5	10/01/2014	05/31/15
	Chapter 6	10/01/2014	10/31/14
	Chapter 1	11/01/2014	11/30/14
	Chapter 7	12/01/2014	12/31/14
	Finalizing the document	01/01/2015	01/31/15

Table A.1: The project schedule

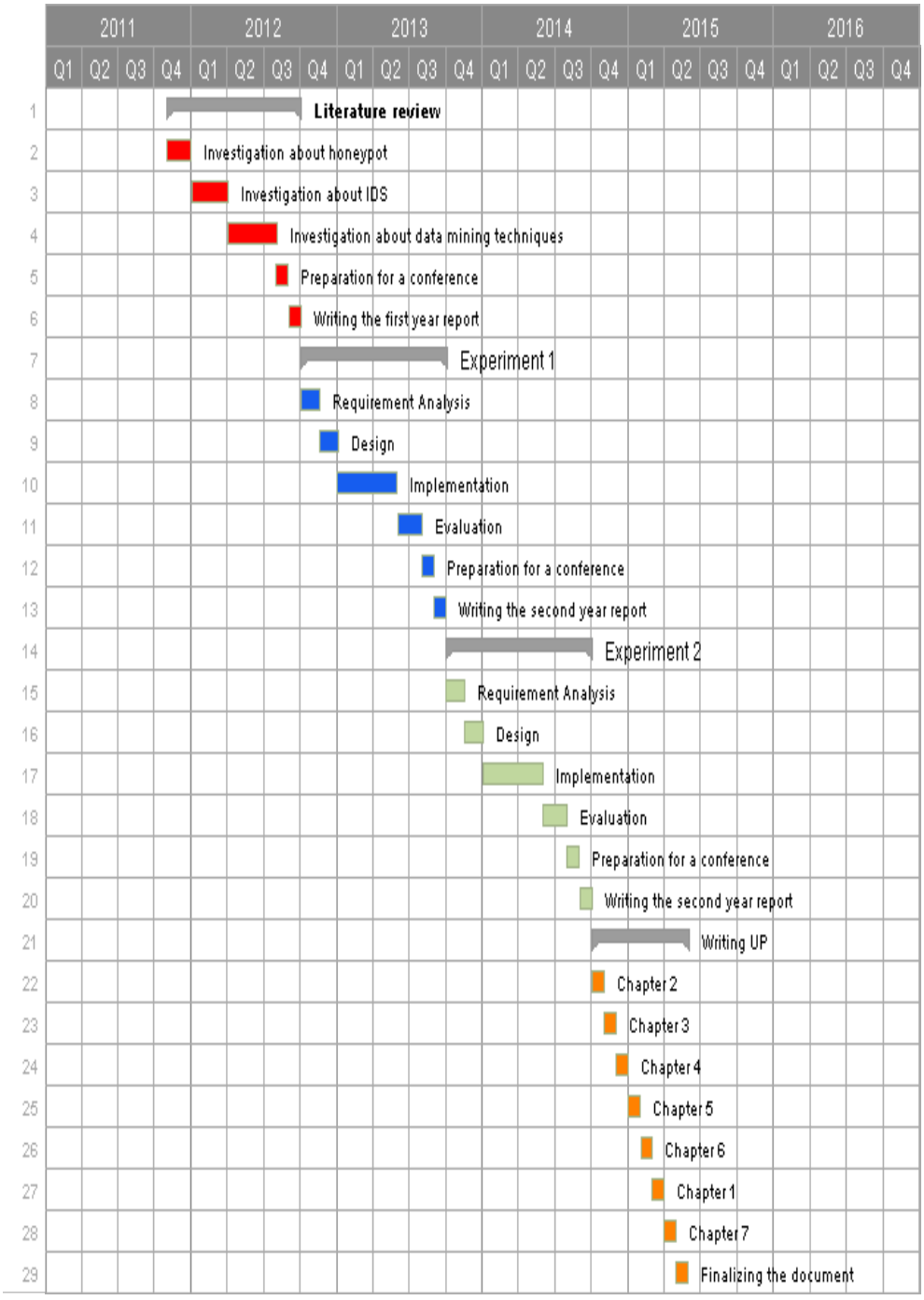

Fig A.1: GANT Chart

A.3 Resource plan

In this research project, some software tools and equipment are required along this research project. Table A.2 shows the schedule of resource.

Resource	Purpose of usage	Start Date	End Date
Software			
Method123	Project Management	01/11/2011	01/08/2015
Wika	Implementation	01/01/2013	01/08/2015
PHP	Implementation	01/01/2013	01/08/2014
MYSQL	Implementation	01/01/2013	01/08/2014
Virtual Machines	Testing	01/01/2013	01/08/2014
Equipment			
Machine with internet connection	Finding information from the internet, writing reports and working in the development environment.	01/11/2011	01/08/2015
Web Server	Testing	01/01/2013	01/08/2014

Table A.2: Resource plan

A.3 Communications plan

Table 1.3 describes the communication activities required to get and receive information in a timely manner.

Activity	Time Frame	Description
Phase Completion Discussion	Yearly	Discussing phase outcomes and plan for following phase (Meeting the supervisor)
Feedback Request	Monthly	Requesting a feedback from the supervisor after completing a piece of work (e.g. conference paper, design)

Table A.3: Communication plan

A.4 Risk plan

A.4.1 Feasibility – Automated Data Analysis

Likelihood	Impact	Priority
High	High	High

Risk Description:

In the project schedule, some activities may be underestimated. Therefore, there is a chance to miss the deadline.

Contingency Plan:

This risk can be overcome by working in holidays and increasing the number of working hours per day to 10 hour in order.

A.4.2 Data Loss

Likelihood	Impact	Priority
High	High	High

Risk Description:

There is a chance to loss all project data and files due to some hardware failure.

Contingency Plan:

It is essential to store the data in multiple device in addition to a cloud storage area.

A.4.3 Feasibility – Automated Data Analysis

Likelihood	Impact	Priority
Medium	Medium	Medium

Risk Description:

In order to perform data analysis, various data mining algorithms would be studied for their feasibility with regard to application in anomaly detection autonomously. This would require initial creation implementation based in offline data that would be subjected to analysis. There is a risk that a rather than one technique, a combination of methods would be useful for analysis.

Contingency Plan:

In order to mitigate this risk it would be ensured that technique or combination of techniques that lead to high true positive rate and low false negative rate would be selected. Based on these results of initial feasibility for a data mining technique with regard to anomaly detection, techniques would be selected for implementation in automated data analysis.

A.4.4 Implementation Issues

Likelihood	Impact	Priority
High	Low	Medium

Risk Description:

Implementation of analysis techniques may require additional set of skills with regard to programming that researcher may have to learn or get trained for. This may cause delay and can have cascading effect on all the following activities.

Contingency Plan:

The researcher aims to use Java for purpose of development. Various third party application programming interfaces have been written for analysis techniques in Java, hence it would mitigate the risk of delay caused due to creation of technique.

Appendix B: Neural Network Training Code

```php
<?php
    //to avoid the script termination
    ignore_user_abort(1);
    set_time_limit(0);

    //including the neural network class
    // the class has been written by Michael Negnevitsky
    //"{@link http://www.amazon.com/link/dp/0321204662
    // Artificial Intelligence - a guide to intelligent systems}"
    require_once ("class_neuralnetwork.php");

    // Create a new neural network with 10 input neurons,
    // 1 hidden layer with 8 neurons, and 1 output neuron
    $n = new NeuralNetwork(10, 8, 1);

    //to load the weights for the neural network we read them from neural.txt
    //if the file does not exits, the weights are set with random values
    $n->load('neural.txt');

    //avoid displaying debugging output
    $n->setVerbose(false);

    // we try training the network for at most $max times
    //number of rounds
    $max = 3;

    $i = 0;
    // train the network in max 1000 epochs, with a max squared error of 0.01
    while (!($success = $n->train(1000, 0.01)) && ++$i < $max) {
        echo "Round $i: No success...<br />";
    }
?>
```

Appendix C: Fuzzy Rules Generation Code

```php
<?PHP
    //to avoid the script termination
    ignore_user_abort(1);
    set_time_limit(0);
    ini_set("memory_limit", 9564217728);

    function get_description($value) {

        if ($value < 0.043) {
            return "VL";
        } elseif ($value >= 0.043 && $value < 0.375) {
            return "L";
        } elseif ($value >= 0.375 && $value < 0.75) {
            return "H";
        } elseif ($value >= 0.75) {
            return "VH";
        }
    }

    //start of the program
    $connection = new mysqli('localhost', 'user', 'user', 'IDS');
    $result = $connection->query("SELECT distinct(data) from training");
    if (!$result) {
        echo "failed to retrieve data";
        exit;
    }

    $max = array(1000, 66, 1000, 3, 1, 511, 1, 1, 1, 1);
    while ($row = $result->fetch_assoc()) {

        $x = json_decode($row['data'], true);
        $data = array($x[0], $x[2], $x[4], $x[7], $x[13],
                    $x[22], $x[34], $x[35], $x[36], $x[37]);
```

```
        //normalization process
        foreach ($max as $key => $m) {
            if ($data[$key] < $m)
                $data[$key] = $data[$key] / $m;
            else
                $data[$key] = 1;
        }

        //get rules
        $rule = array();
        foreach ($data as $key => $item) {
            if ($key == 1)
                $rule[] = "protocol-" . $x[2];
            else
                $rule[] = get_description($item);
        }

        //output
        if ($x[41] == '1') {
            $rule[] = 'N';
        } else {
            $rule[] = 'A';
        }

        $json_data = json_encode($rule);
        $insert = $connection->query(
            "REPLACE INTO rules (rule)  VALUES ('" . $json_data . "')"
        );
        if (!$insert) {
            exit;
        }
    }
?>
```

Appendix D: Hybrid Module Code

```php
<?php

    //to avoid the script termination
    ignore_user_abort(1);
    set_time_limit(0);

    //increase the memory limit
    ini_set("memory_limit", 9564217728);

    //including fuzzy logic class
    require_once ('fuzzy_logic.php');

    //including the neural network class
    require_once ("class_neuralnetwork.php");

    // Create a new neural network with 10 input neurons,
    // 1 hidden layer with 8 neurons, and 1 output neuron
    $n = new NeuralNetwork(10, 8, 1);

    //to load the weights for the neural network we read them from neural.txt
    //if the file does not exits, the weights are set with random values
    $n->load('neural.txt');

    $x = new Fuzzy_Logic();

    //clear any membership functions defined by default
    $x->clearMembers();

    /* ---------- set input members --------- */
    $x->setInputNames(array('feature_0', 'feature_1', 'feature_2', 'feature_3',
                            'feature_4', 'feature_5', 'feature_6', 'feature_7',
                            'feature_8', 'feature_9'));
```

```
for ($i = 0; $i < 10; $i++) {

    if ($i == 1) {

        for ($j = 1; $j <= 66; $j++) {
            // the class does not support singelton membership function.
            // however, we can implement it using LInfinity membership
            // function by setting the three parameters to the same value
            // ($j in our case)
            $name = 'protocol-' . $j;
            $x->addMember($x->getInputName($i), $name, $j, $j, $j, LINFINITY);
        }
    } else {

         $x->addMember($x->getInputName($i), 'VL', 0, 0, .05, LINFINITY);
         $x->addMember($x->getInputName($i), 'L', 0.01, .25, 0.5, TRIANGLE);
         $x->addMember($x->getInputName($i), 'H', .25, 0.5, 1, TRIANGLE);
         $x->addMember($x->getInputName($i), 'VH', .5, 1, 1, RINFINITY);
     }
}


/* ---------- set output members --------- */
$x->setOutputNames(array('OUT'));

$x->addMember($x->getOutputName(0), 'A', 0, .7, .75, LINFINITY);
$x->addMember($x->getOutputName(0), 'N', .7, .75, 1, RINFINITY);

/* ---------- set rule table ------------ */
// store the rules in the fuzzy object
$x->clearRules();

$rules = array();
$connection = new mysqli('localhost', 'user', 'user', 'IDS');
$result = $connection->query("SELECT * from rules");
while ($row = $result->fetch_assoc()) {
    $rules[] = json_decode($row['rule'], 1);
}

$x->setFuzzyTable($rules);
////////////////////////////////////////


$selected_features = array(0, 2, 4, 7, 13, 22, 34, 35, 36, 37);
```

```
//max of each feature used for normalization
$max = array(1000, 66, 1000, 3, 1, 511, 1, 1, 1, 1);

//counter
$counter = 0;

//number of errors
$error = 0;

//number of normal patterns
$number_normal = 0;

//number of false positive (incorrectly classified)
$false_positive = 0;

//number of attack/pattern
$number_threat = 0;

//number of threat patterns detected by the neural network
$detect = 0;


/* test model */
$training = $connection->query(
        "SELECT distinct(data) from validation"
        );


while ($row = $training->fetch_assoc()) {
   //neural part
   $neural_index = $counter;
   $counter++;

   //read inputs
   $neural_inputs = $n->getTrainInputs($neural_index);

   //calculate output
   $calculated = $n->calculate($neural_inputs);

   //convert from array to a single value
   $calculated = $calculated[0];

   //we selected the threshold as 0.7
   if ($calculated > .7)
       $neural_result = 1; //Normal
   else
       $neural_result = 0; //attack
```

```
        //fuzzy part
        //from string to array
        $data = json_decode($row['data'], 1);
        $selected = array();

        //normalize the output
        for ($z = 0; $z < 10; $z++) {
            $index = $selected_features[$z];
            $value = $data[$index];

            if ($z != 1) {
                if ($value > $max[$z])
                    $value = 1;
                else
                    $value = $value / $max[$z];
            }

            //set the input value in the fuzzy logic module
            $x->SetRealInput('feature_' . $z, $value);
        }

        //calculate output
        $fuzzy_arr = $x->calcFuzzyAlt();
        $fuzzy = $fuzzy_arr[0];
        $actual = $data[41];

        //we selected the threshold as 0.5
        if ($fuzzy >= .5)
            $fuzzy_result = 1; //Normal
        else
            $fuzzy_result = 0; //attack
    //the hybrid result , anding between the neural and fuzzy module
    $result = (int) ($fuzzy_result && $neural_result);
    if ($actual == 1) {
        $number_normal++;
        if ($result != 1)
            $false_positive++;
    }
    else {
        $number_threat++;
        if ($result == 0)
            $detect++;
    }
}
```

```
    //storing results in a file result.txt
    $file = fopen("result.txt", "w");
    if ($file) {
        fwrite($file, "Total_number: " . $counter . "\r\n");
        fwrite($file, "Number_normal: " . $number_normal . "\r\n");
        fwrite($file, "False positive: " . $false_positive . "\r\n");
        fwrite($file, "Number_threat: " . $number_threat . "\r\n");
        fwrite($file, "Detect: " . $detect . "\r\n");
        fclose($file);
    }
?>
```

Appendix E: Network sniffing module Code

```
<?PHP
WHILE (1 < 5) {

    //get one packet
   $response = system("tcpdump -i 2 -c 1 -n");

    //Find IP addresses
    $match_patterns = NULL;

    preg_match('/IP([0-9.])+>([0-9.])+/',$response,    $match_patterns);

    //if there is no IP addresses skip this packet
    if (!count($match_patterns)) {
        continue;
    }
    //clean the found patterns
    $filter = preg_replace('/IP /i', '', $match_patterns[0]);

    //get source and destination
    $ip_addresses = explode(' > ', $filter);
    $source = $ip_addresses[0];

    //exclude port
    $source = explode('.', $source);
    unset($source[count($source) - 1]);
    $source = implode('.', $source);
    $destination = $ip_addresses[1];
    $destination = explode('.', $destination);
    unset($destination[count($destination) - 1]);
    $destination = implode('.', $destination);
?>
```

Appendix F: IP Information Finder Module

```php
<?PHP
    /*
     * This function handles the api requests
    */
    function handle_request($method, $ip) {
        //Build the post body
        $post_data = array(  "user-id" => USER_ID, "api-key" => API_KEY, "ip" => $ip );

        //build the url
        $url = "https://neutrinoapi.com/$method";

        //initialize curl and set curl options
        $ch = curl_init($url);
        curl_setopt($ch, CURLOPT_POST, 1);
        curl_setopt($ch, CURLOPT_POSTFIELDS, http_build_query($post_data));
        curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
        curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, false);
        curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);

        //execute the curl request
        $content = curl_exec($ch);

        //format the result the return it
        $result = json_decode($json, true);
        return $result;
    }

    /*
     * This function makes API calls
     */
    function ip_info_finder($ip) {

        //find ip geographic location
         handle_request("ip-info ", $ip);

         //find ip block-list info
         handle_request("ip-blocklist ", $ip);

         //find ip host reputation
         handle_request("ip-host-reputation ", $ip);
     }
?>
```

Appendix G: The reasoning Module Code

```php
<?php

    require_once ('../fuzzy_logic.php');

    //inputs ()
    $inputs = array('is_blacklisted_country' => 1,
                          'is_blocklisted' => 0,
                          'is_anonymous_proxy' => 0,
                          'is_exit_tor_node' => 0,
                          'ip_rating' => 1.2);

    $x = new Fuzzy_Logic();

    $x->clearMembers();
    /* ---------- set input members --------- */
    $x->setInputNames(array('is_blacklisted_country',
                                        'is_blocklisted',
                                        'is_anonymous_proxy',
                                        'is_exit_tor_node',
                                        'ip_rating'));

    for ($i = 0; $i < 5; $i++) {

        if ($i == 5) {
            // define membership function for IP rating
            $x->addMember($x->getInputName($i), 'LOW', 0, 0, 1, TRIANGLE);
            $x->addMember($x->getInputName($i), 'High', 0, 1, 1, TRIANGLE);
        } else {
            // define membership function for other inputs
            $x->addMember($x->getInputName($i), 'SET', 1, 1, 1, TRIANGLE);
        }
    }
```

```
    /* ---------- set output members --------- */
    $x->setOutputNames(array('Result'));
    $x->addMember($x->getOutputName(0), 'Malicious', 0, 1, 1, TRIANGLE);


    /* ---------- set rule table ------------ */
    $x->clearRules();

    // read rules from the database
    $rules = array();
    $this->connection = new mysqli('localhost', '', '', 'multi_stage_rules');

    $result = $this->connection->query("SELECT * from rules");
    while ($row = $result->fetch_assoc()) {
        $rules[] = json_decode($row['rule'], 1);
    }
    $x->setFuzzyTable($rules);


    //apply input to the system
    foreach ($inputs as $name => $value) {
        $x->SetRealInput($name, $value);
    }

    //calculate output
    $fuzzy_arr = $x->calcFuzzyAlt();
    $fuzzy = $fuzzy_arr['OUT'];
?>
```

Appendix H: The Network Sniffing Module Code with a message broker

```php
<?PHP

    WHILE (1 < 5) {

        //get one packet
        $response = system("tcpdump -i 2 -c 1 -n");

        //Find IP addresses
        $match_patterns = NULL;

        preg_match('/IP ([0-9.])+ > ([0-9.])+/', $response, $match_patterns);

        //if there is no IP addresses skip this packet
        if (!count($match_patterns)) {
            break;
        }

        //clean the found patterns
        $filter = preg_replace('/IP /i', '', $match_patterns[0]);

        //get source and destination
        $ip_addresses = explode(' > ', $filter);

        $source = $ip_addresses[0];

        //exclude port from source
        $source = explode('.', $source);
        unset($source[count($source) - 1]);
        $source = implode('.', $source);
```

```
        // exclude port from destination
        $destination = $ip_addresses[1];
        $destination = explode('.', $destination);
        unset($destination[count($destination) - 1]);
        $destination = implode('.', $destination);

        //check whether it is incoming or outgoing packet
        $ip = $source;
        if ($source == NETWORK_ADDRESS) {
            $ip = $destination;
        }

        //send messages
        // create A RABBITMQ connection
        $conn = new AMQPConnection(AMQP_HOST, AMQP_PORT, AMQP_USER,
                           AMQP_PASS, AMQP_VHOST);

        $ch = $conn->channel();
        $ip_msg = json_encode(array('ip' => $ip));

        $activate_msg = new AMQPMessage($json_encoded_object,
                                array(
                                   'content_type' => 'application/json',
                                   'delivery_mode' => true
                                ));
        $ret = $ch->basic_publish($ip_msg, '', 'ip_queue');
    }
?>
```

Appendix I: The IP Information Module Code with a message broker

```php
<?PHP

    //This function handles the api requests
    function handle_request($method, $ip) {

        //Build the post body
        $post_data = array(
            "user-id" => USER_ID, "api-key" => API_KEY, "ip" => $ip
        );

        //build the url
        $url = "https://neutrinoapi.com/$method";

        //initialize curl and set curl options
        $ch = curl_init($url);
        curl_setopt($ch, CURLOPT_POST, 1);
        curl_setopt($ch, CURLOPT_POSTFIELDS, http_build_query($post_data));
        curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
        curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, false);
        curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);

        //execute the curl request
        $content = curl_exec($ch);

        //format the result the return it
        $result = json_decode($json, true);
        return $result;
    }
```

```
    // This function makes API calls
    function ip_info_finder($ip) {

        // find ip geographic location
        handle_request("ip-info ", $ip);

        // find ip block-list info
        handle_request("ip-blocklist ", $ip);

        // find ip host reputation
        handle_request("ip-host-reputation ", $ip);
    }

    //Defining a connection
    $conn = new AMQPConnection(AMQP_HOST, AMQP_PORT,
                       AMQP_USER, AMQP_PASS, AMQP_VHOST);
    $ch = $conn->channel();

    //declaring a queue
    $ch->queue_declare('ip_queue', false, true, false, false);

    //register the consumer
    $ch->basic_consume($queue, "", false, false, false, false, 'ip_info_finder');

    function shutdown($ch, $conn) {

        $ch->close();
        $conn->close();
    }

    register_shutdown_function('shutdown', $ch, $conn);

    // Loop as long as the channel has callbacks registered
    while (count($ch->callbacks)) {
        $ch->wait();
    }
?>
```

Appendix J: The Test Script for Multi-stage Prediction

```php
<?PHP

//include the modules code
require('ip_info_module.php');
require('reasoning_module.php');

//initializing the figures: true positive, true negative,
//faslse positive, false negative
$true_positive = 0;
$true_negative = 0;
$false_positive = 0;
$false_negtive = 0;
$positive_count = 0;
$negative_count = 0;

//open the ip list file
if (($handle = fopen('ip_list.csv', "r")) !== FALSE) {

    //go through each ip address in the list
    while (($data = fgetcsv($handle, 1000, ",")) !== FALSE) {

        //the ip address is in the first column
        $ip = $data[0];

        //the expected result (malicious or normal) in
        //the second column
        $expected_result = $data[1];

        //Apply the ip to the ip info finder module
        $inputs = ip_info_module($ip);
```

```
        //apply the ip info (inputs) to the reasoning module
        $result = reasoning_module($inputs);


        //updaing the figures
        if ($result != $expected_result) {

            // the obtained result is different than the expected
            if ($expected_result == 'Malicious') {
                $positive_count++;
                $false_negative++;
            } else {
                $negative_count++;
                $false_positive++;
            }
        } else {

            // the obtained result is similar to the expected
            if ($expected_result == 'Malicious') {
                $positive_count++;
                $true_negative++;
            } else {
                $negative_count++;
                $true_positive++;
            }
        }
    $count++;
    }

    //printing figures
    echo "True Positive: " . ($true_positive / $positive_count);
    echo "True Negative: " . ($true_negative / $negative_count);
    echo "False Positive: " . ($false_positive / $negative_count);
    echo "False Negative: " . ($false_negative / $positive_count);
}
```