



Faculty of Computer Science and Information Technology

**A Dynamic Approach using Indicators of Compromise to Detect
Malicious Code**

Chiadighikaobi Ikenna Rene

**Master of Science
2018**

A Dynamic Approach using Indicators of Compromise to Detect Malicious Code

Chiadighikaobi Ikenna Rene

A thesis submitted

In fulfilment of the requirements for the degree of Master of Science

(Computer Science)

Faculty of Computer Science and Information Technology
UNIVERSITI MALAYSIA SARAWAK
2018

DECLARATION

I hereby declare that the thesis is based on my original work except for quotations and citation, which have been duly acknowledged. The thesis has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

Name: Chiadighikaobi Ikenna Rene

Matric No. 16020033

Faculty of Computer Science and Information Technology

Universiti Malaysia Sarawak

ACKNOWLEDGEMENT

First, I would like to thank, praise and give glory to the Almighty God for the grace, love and favor He bestowed on me during this program. This journey was encouraged by the wonderful family God gave me, they showed their support through love, provision, financial assistant and most importantly they laid the foundation for the progress. I like to extend an appreciation to my Supervisor, Assoc. Prof. Dr. Johari Bin Abdullah for his guidance and his guidance. I would like to appreciate all my friends and everyone who showed love and care to me during the course of this journey.

ABSTRACT

Malicious activities (malcode) are self replicating malware and a major security threat in a network environment. Timely detection and system alert flags are very essential to prevent rapid spreading of malcode in the network. Automatic signature generation systems has likewise been use to address the issue of malcode, yet there are many works required for good detection. Based on the behavior way of malcode, a behavior approach is required for such detection. In this thesis a dynamic approach technique is proposed for malcode detection and rapid malcode behavior rules are automatically generated based on their Indicator of Compromise (IOC) behavior, as this approach is achieved using Weka system for clustering technique, T-Pot for intrusion data collection, Cuckoo Sandbox for malware data analysis and OpenIOC for IOC creation The experimental study in this thesis highlights the weakness in Signature-Based detection and static analysis of malcode data. The experimental study shows that the proposed approach using IOCRule achieved a detection rate of 87.50%, false negative of 12.50% when evaluated using CTU 2016/2017 Malware dataset. As the evaluation of CTU 2016/2017 Malware dataset achieved a detection rate of 1.18% and a false negative rate of 98.82%. This shows that the proposed approach achieved a much higher detection rate and lower false negative rate compared to the signature-based detection.

Keywords: Malware, indicators of compromise, IDS, malcode, dataset, honeypot.

Pendekatan Dinamik Menggunakan Petunjuk Kompromi untuk Mengesan Kod Berbahaya

ABSTRAK

Aktiviti berniat jahat (malkod) adalah malware yang menduplikasikan diri dan merupakan ancaman kepada keselamatan utama dalam rangkaian persekitaran. Pengesanan yang tepat pada masanya dan sistem amaran sangat penting untuk mencegah penyebaran malkod yang pantas dalam rangkaian. Sistem penjanaan tandatangan automatik juga telah digunakan untuk menangani isu malkod, namun terdapat banyak langkah yang diperlukan untuk pengesanan yang baik. Berdasarkan tingkah laku malkod, pendekatan tingkah laku diperlukan bagi pengesanan sedemikian. Di dalam tesis ini, teknik pendekatan yang dinamik dicadangkan untuk pengesanan malkod dan peraturan tingkah laku malkod yang cepat dijana secara automatik berdasarkan tingkah laku Indikator Kompromi (IOC), kerana pendekatan ini dicapai dengan menggunakan sistem Weka untuk teknik Kluster, T-Pot untuk pengumpulan data pencerobohan, Cuckoo Sandbox untuk analisis data malware dan OpenIOC untuk penciptaan IOC. Kajian eksperimen di dalam tesis ini menunjukkan kelemahan dalam pengesanan berasaskan Tandatangan dan analisis statik data malkod. Kajian eksperimen menunjukkan bahawa pendekatan yang dicadangkan menggunakan IOCRule mencapai kadar pengesanan 87.50%, negatif palsu pada kadar 12.50% ketika dinilai menggunakan dataset CTU 2016/2017 Malware. Sebagai penilaian CTU 2016/2017, set data Malware mencapai kadar negatif palsu 98.82%. Ini menunjukkan pendekatan yang dicadangkan mencapai kadar pengesanan yang jauh lebih tinggi dan kadar negatif palsu yang lebih rendah berbanding pengesanan berasaskan tandatangan.

Kata kunci: *Malware, petunjuk kompromi, IDS, malkod, dataset, honeypot.*

TABLE OF CONTENTS

	Page
DECLARATION	i
ACKNOWLEDGEMENT	ii
ABSTRACT	iii
ABSTRAK	v
TABLE OF CONTENTS	vii
LIST OF TABLES	xi
LIST OF FIGURES	xii
LIST OF ABBREVIATIONS	xiv
CHAPTER 1: INTRODUCTION	1
1.1 Research Background	1
1.1.1 Network Security	1
1.1.2 Malware Analysis	3
1.2 Problems Statement	5
1.3 Objective	7
1.4 Scope and Limitation	7
1.5 Significance of this Research	7
1.6 Thesis Outline	9
CHAPTER 2: LITERATURE REVIEW	10
2.1 Introduction	10
2.2 Intrusion Detection System (IDS)	10
2.2.1 Host-Based Intrusion Detection Systems (HIDS)	12
2.2.2 Network-based Intrusion Detection Systems (NIDS)	13

2.2.2.1	Signature-based Detection	14
2.2.2.2	Anomaly-based Detection	15
2.3	Honeypots and T-Pot System	16
2.3.1	Low-interaction Honeypots	18
2.3.2	High-interaction Honeypots	19
2.3.3	Hybrid Honeypot	19
2.3.4	Honeypot Implementation Strategies	20
2.3.5	T-Pot	21
2.4	Machine Learning	22
2.4.1	Supervised Machine Learning	22
2.4.2	Unsupervised Machine Learning	23
2.5	Malicious code (Botnet)	24
2.5.2	Botnet Life Cycle	25
2.6	Indicators of Compromise (IOC)	26
2.7	Related Works	28
2.7.1	Signature Generation using String Algorithm	28
2.7.2	Malware Analysis and Detection	29
2.7.3	Signature Generation and Intrusion Detection using Machine Learning	31
2.7.4	Clustering and Classification on malware data	33
2.7.5	IOC as a detection technique	34
2.8	Weakness of Reviewed Work	43
2.9	Summary	43

CHAPTER 3: FRAMEWORK AND METHODS	44
3.1 Introduction	44
3.2 Framework Design	44
3.2.1 Intrusion Detection System (ID)	45
3.2.2 Attack Packet Collection System	47
3.2.2.1 T-Pot Honeypot System	48
3.2.2.2 Architecture Details	50
3.2.3 Packets Behavior Analysis	50
3.2.3.1 Data analysis	50
3.2.3.2 IOC Clustering	51
3.2.3.2.1 Hierarchical	51
3.2.3.2.2 K-Means	52
3.2.3.3 IOC-Writer	54
3.3 Evaluation Parameters	55
3.3.1 Experimental Study	56
3.3.1.1 Experiment 1: IOCRule with Different Clustering	
Algorithm	56
3.3.1.2 Experiment 2: Vary number of IOCRule	57
3.3.1.3 Experiment 3: IOCRule with and without Clustering	
Algorithm	58
3.4 Summary	59
CHAPTER 4: RESULTS AND ANALYSIS	60
4.1 Introduction	60
4.2 Experimental Platform System Architecture	60

4.2.1 Hardware	61
4.3 Formulate Hypothesis	62
4.4 Comparison of Existing Work and Proposed Approach	62
4.4.1 Existing Approach Evaluation with Darpa Dataset	65
4.4.2 Existing Approach Evaluation with CTU Dataset	67
4.4.3 Existing Approach Study Discussion	68
4.5 Proposed Framework Evaluation	69
4.5.1 Information Gathering	70
4.5.2 Sandbox Data Analysis	71
4.5.3 IOC Extraction	72
4.5.4 IOC Clustering	74
4.5.4.1 Hierarchical Clustering	75
4.5.4.2 Kmeans Clustering	76
4.5.4.3 IOC Clustering Result Discussion	77
4.5.5 IOC Writer and IOCRule Creation	77
4.6 Experimental Study to Evaluate IOCRules	78
4.6.1 Experiment 1: IOCRule Using Different Clustering Algorithm	79
4.6.1.1 Experimental Study Using IOCRule achieved from Hierarchical Clustering	79
4.6.1.2 Experimental Study Using IOCRule achieved from K-Means Clustering	80
4.6.1.3 K-means IOCRule Result Discussion	81
4.6.2 Experiment 2: Vary Number of IOCRule	82
4.6.2.1 Detection Rate	83

4.6.2.2	Detection Time	84
4.6.2.3	False Negative Rate	85
4.6.2.4	Experiment 2 Result Discussion	86
4.6.3	Experiment 3: IOCRule with and without Clustering Algorithm	87
4.6.3.1	Experiment 3 Result Discussion	87
4.7	Comparative Result Discussion Between the Baseline Study and the Experimental Study	88
4.7.1	Discussion on each Parameter	89
4.7.1.1	Detection Rate	90
4.7.1.2	Detection Time	91
4.7.1.3	False Negative	91
4.8	Result Conclusion	92
4.9	Summary	94
CHAPTER 5: CONCLUSION AND FUTURE WORK		95
5.1	Summary of Contributions	96
5.2	Challenges	97
5.3	Future Work	98
REFERENCES		99
APPENDICES		111

LIST OF TABLES

	Page
Table 2.1 Advantages and Disadvantages of Honeypots	20
Table 2.2 Related Works	40
Table 3.1 IOCRule with different Clustering Algorithm	57
Table 3.2 Vary Number of IOCRule	58
Table 4.1 Existing Approach Evaluation	69
Table 4.2 Comparison of Hierarchical And K-Means Clustering Result	77
Table 4.3 IOCRule with different Clustering Algorithm	82
Table 4.4 Vary number of IOC	83
Table 4.5 IOCRule with and without Clustering Algorithm	87
Table 4.6 Experimental Result	89

LIST OF FIGURES

	Page
Figure 2.1 IDS Taxonomy	11
Figure 2.2 Botnet Lifecycle	25
Figure 2.3 Example of IOC	27
Figure 3.1 Research Framework	44
Figure 3.2 Snort System Architecture	45
Figure 3.3 Snort detection alert	46
Figure 3.4 Example of Snort Rule	47
Figure 3.5 T-Pot Architecture	49
Figure 3.6 Indicators Example	54
Figure 4.1 Experimental Platform System Architecture	60
Figure 4.2 Comparison Evaluation Flow	64
Figure 4.3 Evaluation result of Darpa Dataset	66
Figure 4.4 Evaluation result of CTU Dataset	67
Figure 4.5 Evaluation Framework Flowchart	70
Figure 4.6 Malware Data	71
Figure 4.7 Data Analysis Result	72
Figure 4.8 Extracted IOC	74
Figure 4.9 Dendrogram (Tree)	76
Figure 4.10 IOCRule Sample	78
Figure 4.11 Evaluation Pseudocode	78
Figure 4.12 Experimental result of CTU Dataset on IOCRule (Hierarchical Clustering)	79

Figure 4.13	Experimental result of CTU Dataset on IOCRule	80
Figure 4.14	Detection Rate	84
Figure 4.15	Detection Time	85
Figure 4.16	False Negative Rate	86
Figure 4.17	Detection Rate	90
Figure 4.18	Detection Time	91
Figure 4.19	Number of False Negative	92

LIST OF ABBREVIATIONS

IOA	Indicators of Attack
IOC	Indicators of Compromise
IDS	Intrusion Detection System
IOCRule	Indicators of Compromise Rule
Malcode	Malicious Code
ML	Machine Learning
NIDS	Network Intrusion Detection System

CHAPTER 1

INTRODUCTION

1.1 Research Background

Networking occurs when there is an interconnection of two or more computers for communication purposes. Communication can be in many forms which includes the distribution of data from one device to another. This data or information is distributed within networks or from one network environment to another. The Internet, also called "the Net", is a connection of networks, where users at any networked device, if permitted, can receive and transmit information from any network point. Due to the amount of data and information available over the network, data theft has become part of the network activity, as network intrusion occurs on daily basis [1].

1.1.1 Network Security

Network security is an essential part of computer networks, as intrusion attack is a threat to network communication. Network intrusion happens when an unauthorized system or user gain access into a network system and manipulates data or information. "There are a total of 184 billion exploited detections, 1.8 billion average daily attack volume, 6,298 unique exploited detections and exploited volume per firm averaged at 2.5 million, with a median of 456, and 69% of firms saw severe attacks" [2]. Network security is an important aspect of networking, because a malicious code (malcode) attack is a threat to the network user.

Malcode is an application security threat that can be in the form of software system and cannot be efficiently controlled by traditional anti-virus software's and systems.

Network intrusion has emerged from known attack to unknown attack (Malcode), as there has been an exponential development in the Malcode family that the present security approach is not able to detect [3, 4]. Over the years, detection of Malcode or malware attack has been done using signature-based approach, whereby different techniques have been applied such as:-

Automating the generation of malware signatures using honeypot, Signature generator and by clustering or classifying the generated signatures and applying the honey pot data with machine learning techniques to generate signatures for intrusion detection [5-9].

Malcode utilizes sophisticated ways to make itself hidden from intrusion detection systems such as anti-malware software and infrastructures. Some malware activities remain undiscovered for years as they steal confidential data and also damage the system [10]. Malcode uses the Internet to call-back-home to communicate with the attack initiating server to receive new tasks and updates. When malcode like Bot-net tries to communicate with its Command and Control (C&C) centre, it uses a known network protocol to pass through network defenses' measures such as firewalls and anti-virus. Malicious programs are capable of hiding themselves or disabling their activity when they detect an attempt to discover them. Therefore, there is a need to use techniques that can detect malicious activities on systems. Some previous works have been done using the static approach of data analysis, as some of the research centered on analyzing network traffic as they focused on network layers and protocol, system files, file structures or on certain malware family [11, 12].

1.1.2 Malware Analysis

There are two general ways to deal with malware analysis, namely static analysis that studies the malware without executing it and dynamic analysis where the behavior of the malware is observed. As malware quickly advances the need for a liable detection system is crucial. Malware may include scanning abilities to the point that each infected host can additionally extend the botnet by exploiting obscure vulnerabilities in operating frameworks. In fact, even with an installed and newly refreshed anti-infection programming, the normal client could remain unnoticed since malware utilizes strategies to remain undetected. The test with obfuscation techniques was displayed in a Black Hat gathering [13], where it was expressed that detecting these sorts of malware is exceptionally troublesome continuous or posthumous analysis. Despite the fact that if the original malware is identified by hostile to infection application, an alternate variation will sidestep the regular example matching system since it yields an alternate example. It is moreover essential to gain information about their behavior to create exact identification plans.

Malware analysis is a technique that helps obtain information about a malware's behavior [14]. Regularly utilized approaches are static analysis that reviews the malware without executing it, and dynamic (behavioral) analysis which examine malware as they execute. Despite the fact that the two strategies may fulfill a similar objective of studying how malware functions, the tools, and aptitudes required are diverse [15]. Static analysis approach is done by analyzing the source code of the malware to understand how it works. Normally, static analysis utilizes string examining tools, such as disassemblers, debuggers and compilers to study source codes. Subsequent to applying these tools on the malware's executable, the investigator or malware examiner goes through the source code to gain

information on how the malware works for instance how it infects systems and transmits. The most common method for conducting a dynamic analysis is to run the malware and see what happens. Note that this approach is not without issues, since you may wind up destroying all information on your framework or let the malware spread if the relinquished host is associated with the Internet.

This thesis focus on malware data analysis, Indicators of Compromise (IOC) feature extraction, clustering and dynamic rules development for malicious code detection. This research is an improved version of the traditional intrusion-detection method, which deals with a static based signature created with String techniques [5, 16], as this research applies Machine learning techniques to cluster malcode behavior together for better IOC creation for intrusion detection. Indicators of compromise referred to as IOCs, consist of one or more artifacts that relate to a particular security incident or attack [17, 18]. The intent of assembling IOCs for a specific item of malware or malicious code is to state, with a relatively high degree of confidence, whether or not such items are present in a given environment. Based on the proposed research, IOC will be used to create a behavior rule for Network Intrusion Detection System (NIDS) for the detection of malcode activities. Over the years different type of systems such as Intrusion Detection System (IDS), Intrusion Prevention System (IPS) and Firewall have been implemented and established to address the issue of a security threat. Most of the system addresses known attack, using misuse techniques or (signature) pattern matching techniques. To lure intrusion attack for data collection in this research, honey-pot system will be implemented [19, 20]. In this research, the honey-pot will be deployed in an isolated environment, (a virtual machine), that consists of software components that constantly analyze the system events and record intrusion behavior.

Security should not be an out-of-the-box solution, as careful analysis of the environment at hand is needed before a solution can be provided. It is a step by step process and a thorough understanding of the system and constraints is needed. Intrusion Detection has been a research focus for long and there are still many issues that need to be addressed. As most IDS use static detection method, which is based on string signature [21], malicious code detection have been a challenging issue since it can incorporate a wide range of dynamic techniques, for example, indirect accesses and Trojan steeds. To distinguish conceivable vindictive practices of such malware can be a monotonous and testing assignment.

In other words, they are produced by impersonation or adjustment rather than advancement. A large portion of them had similar characteristics in relation to specific activities such as [21, 22, 23]:- A server connection programmed start-up, Framework registry access, Hostile to infection software turn off, and Bot software overhauls or uninstall. Furthermore, to perform specific behaviors, they have tendency to use the same function calls with a different structure. From the characteristics, the mining of similarity among system calls from many bot binaries derives the common semantic behavior to represent entire malicious code family for malware detection. The goal of this work is to identify a (metamorphic) Malcode using IOC and IOC-based rules detection method, incorporated with behavior analysis.

1.2 Problems Statement

According to Kaur and Singh [24], "the best intrusion attacks that keep away from discovery are Malcode as they do not demonstrate particular behaviors". Signature IDS lack the capability to detect Malcode (Unknown attack). The aim of this thesis is to detect

Malcode attack and identify attack features for intrusion IOCRule generation. Malcode attacks are vulnerabilities that are unpredictable at the time of attack, and it has high negative impact to the network environment [25, 26, 27].

Current research focus on improvement of intrusion signature generation for Intrusion Detection, as Signature-based Intrusion detection technique faces a challenge in high accuracy detection, due to its approach of detection [38, 67, 68, 71]. This technique uses static approach and intrusion signatures must exist in the Detection System database in order to enable detection of attack.

Signature-based Intrusion Detection System (IDS), which can be referred as pattern matching technique has been used by many organizations in detecting network threat, but it is unable to detect unknown (Malicious Code) attack [28]. Therefore there is a need for dynamic approach of detection rule to address the issue of unknown attack detection using Signature-based IDS [29, 30, 31].

Signature-based IDS focus its detection techniques on signatures, derived from static data analysis, which makes it difficult for malcode detection [5]. In order to have a good detection mechanism or outcome, the data analysis process plays an important role. [32, 33, 34]. This research uses dynamic data analysis to overcome the problem of static analysis. Malware analysis is a procedure to perform analysis of malware and study the behavior of malware [35, 36]. Many works have been done in malware analysis as many uses static methods, which is a method of malware analysis done without running the malware [13, 28, 37, 39].

1.3 Objectives

The main objective of the research is to design and develop a high accuracy Network Intrusion Detection method for malicious code activities. Other objectives of the research are as follows:

- a. To design an algorithm to detect unknown network attack.
- b. To analyze intrusion data and extract IOC from analyzed data.
- c. To develop IOCRule for malicious code detection.

1.4 Scope and Limitation

The research is designed to aid existing intrusion detection methods, with the additional capability to detect variations of network intrusions in the form of malicious codes. To enable behavior analysis, IOC techniques are used to investigate and detect variants of known and unknown attacks while they are being transferred over a network. The similarity between familiar malicious code samples and unfamiliar incoming traffic is calculated based on related information or related symbols [14]. In this thesis, the research scope is restricted to intrusion-detection schemes whereby intrusion log can be used to perform packet analysis and prepare a result for future malware detection based on IOCRule. This research applies behavioral approach in intrusion rule's generation and to reduce false alarm it uses machine learning for feature clustering used for rules generation, which is a dynamic algorithm and improve detection rate.

1.5 Significance of the Research

This research explored methods and techniques of existing measures for honeypot based Malcode data collection, behavior analysis using Cuckoo Sandbox and identifying

their shortcomings. The aim of this research is to develop a conceptual approach for advanced, integrated malware collection, IOC as an approach or method to identify malicious code and IOC rule base generation for intrusion detection. To accomplish this task the procedure is divided into several steps:

First, the state of the art in a honey-pot based malware data collection and analysis is outlined in brief. Based on these findings a holistic approach for integrated malware collection, analysis malware data, IOC extraction and overcoming the shortcomings is described in a second step. Next, a proof of concept implementation of the service emulation part as one component of the approach is done in a further step. The focus is on handling unknown traffic patterns, such as malicious code. Finally, the feasibility of the proof of concept is validated using a testing data set to prove the IOC base rules generated for the improved intrusion-detection method, which is the key contribution of this thesis. The assumption of our training data set is that it is able to follow the whole malware life-cycle to enable a better understanding of currently ongoing or emerging attacks by malicious code. The findings are expected to make a significant contribution to the enhanced analysis of current malware and malware intrusion method in this research.

The outcome of this research ensures high-accuracy intrusion detection, which is achieved by engaging in behavior approach of malware data analysis and dynamic rule development. High-accuracy intrusion detection is one of the significance of this research, as it will overcome the one major problem of signature-based IDS. Signature-based IDS is unable to offer good detection rate, as it uses static techniques.

This research implements a dynamic algorithm to detect novel or unknown attack, as this approach overcome the problem of signature-based IDS, because it uses the string

algorithm, which can only generate static approach of detection. The algorithm is used to cluster IOC features extracted from the analyzed malware data. The purpose of this algorithm is to cluster behavior of malware, rather than String algorithm which signatures for traditional IDS uses.

1.6 Thesis Outline

The remainder of the thesis is organized as follows:

Chapter 1 discusses the general view of the research, and introduce the objective and aim of the research.

Chapter 2 discusses the literature review, which elaborates on the research related works, explaining approaches existing or similar works used.

Chapter 3 explain the framework of this thesis, the methods, approaches and experimental steps that is used to achieve the objective and aim of this research.

Chapter 4 states the Evaluation or Experimental study and Result discussion of the research. It shows the experimental study and results achieved and the conclusion of the study.

Chapter 5 concludes the thesis and states the Future Work of the research.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This chapter reviews the state of the art in intrusion-detection technologies, the history, types, strengths, and weaknesses. It involves the description of knowledge and gaps, which occurred in previous and related research. Its purpose is to summarize, synthesize and analyze the arguments of others. Besides that it reveals the similarities and differences, consistencies and inconsistencies and controversies in previous research and introduces computer or network intrusion, discusses the need to study them for clustering and IOCRule base detection purposes. Data collection methodologies for benchmarking intrusion-detection systems and approaches to extract and create Indicator of compromise are studied. honeypot technologies are presented in this thesis as useful tools for data collection and attack analysis [19].

2.2 Intrusion Detection System (IDS)

The Internet Security Glossary (RFC2828) defines an “Intrusion” as [32, 40] “A security event, or a combination of multiple security events, that constitutes a security incident in which an intruder gains, or attempts to gain access to a system (or system resource) without having the authorization to do so.”. The use of sophisticated means to exploit vulnerable system has been employed by intruders and attackers to:

Access unauthorized system over the network or internet remotely, Gain unauthorized additional user privileges on a system and Misuse privileges granted for malicious gain.

IDS enables organizations and enterprises to protect their systems from the onslaught of attackers or malicious codes [41, 42]. On detecting an attack or intrusive behavior, IDS can be configured to respond, either by logging details of the attack and the attacker or by actively taking automated actions to further mitigate the spread of the attack [43, 44]. Careful consideration needs to be taken while configuring IDS. Bad or misconfiguration of IDS may lead to either too many alarms or false positives (FP), or undetected attacks or False Negatives (FN). Intrusion detection is not a plug-and-play solution as highlighted Judy Novak as “Intrusion detection is not a specific tool but a capability, a blending of tools and techniques” [6, 45]. It requires expert level knowledge, skill, and experience to install, configure and maintain such systems in a home or enterprise environment.

Figure 2.1 shows the taxonomy of IDS and its detection techniques. An IDS is a system that monitors network or systems for malware activities and produces reports to an administration station [46]. This thesis is focusing on the signature-based technique of network intrusion detection system.

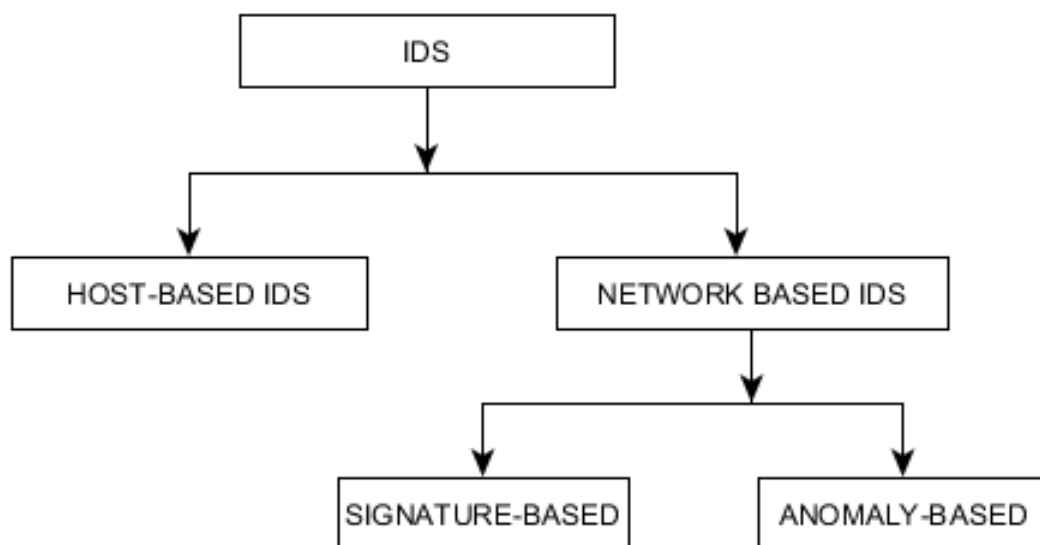


Figure 2.1: IDS Taxonomy

IDS detect attacks and misuse of information either by legitimate users of the system or external to exploit security vulnerabilities as mention by Teresa et al, [46]. This detector can launch probes to trigger the audit process, such as requesting version numbers for applications [7, 47]. In this research we will implement IOCRule into IDS for detection purposes. The generation of IOCRule is done by analyzing malicious dataset using cuckoo sandbox and to extract IOC (features). IOCs are identities of intrusion behaviors in a system or network that states with high degree of confidence of the type of attack or intrusion.

With the increasing number of systems in a network environment, it has become necessary to consider IDS not only in single hosts but in the whole networks. Besides Host Intrusion Detection System (HIDS) and Network Intrusion-Detection Systems (NIDS), there are many derivations such as centralized-host-based intrusion-detection system (CHIDS) and Hybrid Intrusion-detection Systems. For the purpose of this research, we will look into network-based IDS, as it is easy accessible (Open Source), unlike Host-based IDS which will require a purchase of physical hardware system. Figure 2.1 shows the types of IDS. The main function of IDS is to monitor and detect unapproved users from getting into a private system associated with the Internet, particularly intranet network and IDS present a log file to the administrator for further action in the network attack [47].

2.2.1 Host-Based Intrusion Detection Systems (HIDS)

HIDS initial design was for mainframe computer, and all users were local network users [47]. The HIDS look for signs of intrusion on the local host system. These frequently use the host system's audit and logging mechanism as a source of information for analysis [24]. They look for unusual activity that is confined to the local host such as logins,

improper file access, unapproved privilege escalation, or alterations on system privileges [10][13]. Another feature found in many HIDS is the hash of the file system. HIDS creates an initial database with hash values of files that are considered important. To identify any violation of recorded data, the hash value is recalculated and compared with correct one previously saved in the database. Among many benefits of using a HIDS, is as follows [14, 15, 45]:

Establishing a strong association between programs and users: identifying the author and the time at which application or commands were executed, because of being one of the potential targets of attack, HIDS is capable of collecting considerable and valuable information during the attack and confronting coming threats directed only to the machine where the HIDS resides without the necessity of capturing whole network traffic. Among the disadvantages of using a HIDS, we can cite the following [45]: High complexity in management and scalability of HIDS (because of the necessity of configuration and constant observation on all hosts, impacting scalability), Operational system dependent, and Susceptible to attacks, resulting in corrupted data or data deletion. This is due to the fact of its priority in direct access and belonging to or being part of the host environment.

2.2.2 Network-based Intrusion Detection Systems (NIDS)

The NIDS can monitor data collected from its own network segment or from multiple network hosts. There are many available NIDS tools including Snort, Bro, PHAD, NADIR and DIDS [82,87]. Many of these tools have data collecting facilities implemented in them. NIDS is intrusion-detection systems that capture data packets traveling on the network media (cables, wireless) and matches them to a database using signatures or anomaly. NIDS is placed in key areas of network infrastructure and monitors traffic as it flows to other hosts. Network based IDS has grown in popularity and outpaced the

acceptance of HIDS [45]. A NIDS is more cost-effective than HIDS because it can protect a substantial portion of network infrastructure with one device.

With NIDS, the intrusion analyst has a wide-angle view of what is happening in and around the network. NIDS can detect many types of attacks such as Denial-Of-Service and Penetration by legitimates users. The role of the network IDS is to flag and sometimes stop an attack before it gets to information assets or causes damage. NIDS is effective for monitoring both inbound and outbound network traffic. Some benefits of using the NIDS includes, No benefit on network performance, Attacks can be identified in real time, thus enabling the administrator to respond to the attacks quickly, Able to capture suspicious attacks that have not yet been identified, and Operational system independent [45].

2.2.2.1 Signature-based Detection

Signature based IDS uses set of static or string rules to detect intrusion. When the existing signature matches with the signatures of incoming attacks, the IDS will log alerts [45,48]. This technique is the most accurate technique of detecting known attacks as it matches incoming traffic to detect known signatures in the database and once identified it craft an alert. In addition, almost every type of malicious traffic can be identified by a unique signature. Therefore, most malicious traffic can be caught by an IDS using signature detection. Signature detection does have some limitations; hence NIDSs that use signature detection are likely to generate false positives and slow performance. According to Farzaneh et al, after an experimental study on how to improve the performance of Signature-based techniques, where the study proposed a solution of using two signature based network intrusion detection systems running Snort in parallel with a portion of packets and a subset of rules, and distributing the traffic between them, the processing time of the traffic will be reduced. Consequently, the performance of the system will be

improved [48][49]. Many research and industry have used many techniques in IDS to detect intrusion packets. Many researches have used machine learning in intrusion detection, as some previous work shows 89% detection rate of zero day attack using machine learning, and over 18 different experiments, show that the ML sensor measured to have 94% detection rate of malicious traffic and false alert rate of 1.8% [50]. Due to the dynamic of treats in networking today, signatures are needed to be advanced enough in other to be useful in detection of attacks. This research will play a vital role to achieve this, improved IDS detection technique.

Snort Signature-based IDS Tool is an open-source network detection and prevention system. It has the capacity to analyze real-time traffic and data flow in the network. It is small, lightweight open source IDS written by [51]. It is configured as an Intrusion Prevention system for monitoring and Intervention of security attacks on networks. This type of IDS can be installed on many platforms such as Linux, Windows, and Mac. It checks the packet against the rule written stored in the database. The snort rule can be written in any language, and its structure can be modified. As a signature-based IDS system, if pattern matches, then the attack can be found but when a new attack like unknown malicious code comes into the network, it is unable to detect it, as the intrusion signature is not in the database. Snort is made up of multiple components and all the component work together in order to detect a specific attack and take the corresponding action that has been required for that precise attack.

2.2.2.2 Anomaly-based Detection

Anomaly-based IDS detect the abnormality in the computer networks [31, 52]. Anomaly is said to occur when there is deviation from a normal behavior to abnormal behavior of packets. Anomaly based IDS detect attacks that it has previous knowledge of;

that is having a pre training of the anomaly-based IDS with packets, which gives it knowledge of the packets behavior for future detection of such packets. Anomaly-based IDS detect intrusion by comparing the new traffic with the already created attack profile. It can be used at the application level to monitor the activity of users. This type of IDS gathers a set of data from the system activity of the user. If the user deviates from the normal usage pattern, an alarm is raised. Anomaly detection can be used to monitor and set up privilege escalation attacks. If a normal user account does not have privileged access to an important operating system file but is seen to access, the IDS determine that potentially damaging activity has taken place and generates an alert [47]. The key benefit of anomaly detection IDS is that they do not need to have previous knowledge of an attack. It just needs to determine that the behavior differs from the normal [52]. Like signature detection, anomaly detection has some limitations as well. You must assume that the data collected in the existing approach dataset is not malicious and is normal activity. If an intruder steals company secrets every night at 2:00 am when the IDS were gathering existing approach data, it would assume that this was normal behavior and never raises an alarm. In this aspect, anomaly detection IDS is prone to false negatives [58].

2.3 Honeypots and T-Pot System

A honeypot is a computer system deliberately vulnerable to one or more threats, deployed in a network for the reason for logging and examining attacks on the honeypot. These systems may be made intentionally weak in protection in order to lure attackers to attack the system. Honeypots are imitation computer assets set up for the reason for monitoring and logging the exercises of elements that test, or them. Exercises on honeypots can be considered suspicious by definition, as there is no point for benevolent clients to interface with these systems [70, 89]. It can, be characterized as "an information system

asset whose worth lies in the unauthorized or illegal utilization of that asset". Basically, a honeypot is a distraction, put out on a network as a snare to lure attackers. Honeypots are ordinarily virtual machines, intended to copy genuine machines, pretending or making the presence of running full administrations and applications, with open ports that may be found on a run-of-the-mill system or server on a network. Honeypots can be utilized for creation or exploration purposes. Most generation honeypots are limitations of particular working systems or administrations [19]. They ensure a network and systems against attacks generated by automated tools used in haphazardly search for and assume control vulnerable systems. Some creation honeypots can even close down attacks, by and large, by, for instance, sending the attackers an affirmation bundle with a window size of zero. This puts the attack into a "hold up" status in which it could just send information when the window size increments. Along these lines, generation honeypots are frequently utilized as reconnaissance or deterrence tools. Research honeypots are real operating systems and services that attackers can interact with, and therefore, involve higher risk. They collect extensive information and intelligence on new attack techniques and methods and hence provide a more accurate picture of the types of attacks being perpetrated. They also provide improved attack prevention, detection and reaction information, drawn from the log files and other information captured in the process. In general, honeypot research institutions such as universities and military departments will run research honeypots to gather intelligence on new attack methods. Some of the research results are published for the benefit of the whole community [53].

Honeypots have been used by many researchers as an intrusion-detection system, not just to collect intrusion data but also to detect intrusion. Most researchers employed Honey pots to capture anomalous streams and generate signatures [45, 54, 55]. Honeypots serves

as an excellent data collection resource to capture malicious streams for analysis. Reshma and Chirag [55], implemented a honeyd honey-pot to detect an intrusion attack. The detected logs are passed to Honeycomb, which is a host-based intrusion-detection system that automatically creates signatures, that applies the Longest Common Substring (LCS) algorithm on the packet content of a number of connections going to the same services. Honeycomb is well suited for extracting string signatures [56]. The Honeycomb was implemented as an unfiltered cable modem connection for three days. The Honeyd setup for the required system has the following services running on the system: Fake FTP, Telnet, SMTP, HTTP services, all Perl/Shell scripts.

In this thesis, we proposed the use of honeypot system to collect botnet attack data and apply IOC, which is part of the feature selection and machine learning algorithm to the log for future intrusion detection and signature generation. Honeypots can be classified into two, which are low-interaction network components like pre-configured traffic sinks, or full-interaction hosts with real operating systems and services. This additional insight helps researchers quickly devise defense mechanisms to fend off any new or novel attacks. They serve as a means to spy on malicious attackers, their organization, their tools and their sophisticated techniques. Since all traffic received by the T-Pot or Honeypot is considered malicious, it becomes easy to focus on attack data compared to gateway devices, which cannot provide these many granularities due to the scale or high volume of traffic they process.

2.3.1 Low-interaction Honey pots

Low-interaction honeypots work by copying certain administrations and system privilege and constrained interaction. The attacker's exercises are constrained to the level

of imitation given by the honeypot. For instance, a copied FTP administration listening on a specific port may just copy an FTP login, or it may further support an assortment of extra FTP commands. The benefits of low-interaction honeypots are that they are straightforward and simple to convey. Furthermore, the restricted copying accessible and/or permitted on low-interaction honeypots diminish the potential dangers achieved utilizing them as a part of the field. On the other hand, with low-interaction honey pots, information can be obtained, and it is conceivable that accomplished attacker's will effortlessly detect a honeypot when they encounter one [57].

2.3.2 High-interaction Honeypots

High-interaction honeypots are more complex, as they include actual working systems and applications. For instance, an actual FTP server will be manufactured if the point is to gather information about attacks on a specific FTP server or administration. By giving attackers genuine systems to associate with, no limitations are forced on attack behavior, and this permits administrators to catch broad insights about the full degree of an attacker's routines. Be that as it may, it is not inconceivable that attackers may assume control over a high-interaction honeypot system and use it as a stepping stone to attack different systems inside of the organization. Therefore, adequate security measures should be executed accordingly [58]. In the worst case scenario, the network associated with the honeypot may be separated to keep attackers from further entering the network and systems in the honeypot system itself.

2.3.3 Hybrid Honeypot

Hybrid Honeypot is a mixed type of Honeypot, combining features and functionality from both low and high interaction honey-pots. Honeypots can also be categorized by the

way they are deployed in a network. This can include server-side Honeypots, which are deployed on a server or host running services. Malicious code attacks and exploits these services. Honeypots can also be deployed as a client-side Honeypots, which act as clients and interact with a malicious server [59]. Table 2.1 shows the advantages and disadvantages of different honeypot types [28].

Table 2.1: Advantages and Disadvantages of Honeypots

Low-interaction Honeypot	High-interaction Honeypot	Hybrid Honeypot
+ fast	- slow	+ fast
- unable to detect unknown attack	+ detect unknown attack	+ detect unknown Attack
+ resists to time-bomb + handles interaction with attackers	- unable to resist time bomb and can't handle interaction with attackers	+ resists to time-bomb +handles interaction with attackers
+ simple to set up and maintain	- complicated to set up and maintain	- complicated to set up and Maintain

2.3.4 Honeypot Implementation Strategies

To maximize the strengths of honeypots, and minimize the risks involved, deployment should be carefully planned. The following is a set of common honeypot deployment strategies [40]:

Install honeypots close to normal creation servers. The honeypot will probably need to reflect some genuine information and administrations from the creation servers so as to attract attackers. The security of the honeypot can be loosened slightly in order to increase its probability of being compromised. The honeypot can then collect attack-related data. Nonetheless, if a successful attack happens on the honeypot inside of the network, the compromised honeypot machine may be utilized to check for other potential attack in the network by studying and analyzing the honeypot logfile.

Pair every server with a honeypot, and direct suspicious traffic bound for the server to the honeypot. For example, traffic at TCP port 80 can be redirected to a web server IP address as typical, while all other traffic to the web server will be redirected towards the honeypot.

Build a honeynet, which is a network of honeypots that copy and replicate a genuine or fictitious network. This will appear to attackers as though a wide range of genuine systems are accessible on a few distinct stages. A honeynet offers an early warning framework against attacks and gives an excellent approach to break down and comprehend an attacker's aim, by taking a gander at what sort of machines and administrations have been attacked, and what kind of attacks have been led.

2.3.5 T-Pot

T-Pot honeypot is a special kind of hybrid-interaction Honeypot and a multi-Honeypot Platform. T-Pot extends the concept of a single Honeypot to a highly controlled network of Honeypots. A T-Pot is a specialized network architecture configured in a way to achieve the following: [60]

Data Control: It deals with the containment of activity within the T-Pot.

Data Capture: It involves in capturing, monitoring and logging of all threats and attacker activities within the T-Pot.

Data Collection: Captured data is securely forwarded to a centralized data collection point.

This architecture creates a highly controlled network, in which one can control and monitor all kinds of system and network activity. Honeypots are then placed within a network to capture incoming and outgoing network traffic. Acting as a transparent gateway

the Honeypot is undetectable by attackers and serves its purpose by logging all network activity going in or out of the Honeypots. Implementation and design challenges of T-Pot Honeypots relevant to our thesis are discussed in Chapter 3, Section 3.2.2.1.

2.4 Machine learning

Machine learning is a branch of Artificial Intelligence (AI), which deals with the study and creation of developing systems or algorithms that can learn and evolve [36]. A machine learner is trained on labeled data and can then use that knowledge to identify patterns in unlabelled data for classification, and identify new patterns as they emerge. With the rapid development of the Internet services and the fast increasing of intrusion problems, the traditional intrusion-detection methods cannot work well with the complicated intrusions [60]. Therefore, introducing machine learning into intrusion-detection systems to improve the performance has become one of the major concerns in the research of intrusion detection [27, 61]. Machine learning algorithms are now being widely deployed for intrusion detection analysis techniques such as unsupervised machine learning technique. Machine learning algorithms can be categorized as: Supervised learning, unsupervised learning and Semi-supervised learning [2, 62].

2.4.1 Supervised Machine Learning

Supervised learning is based on training a data sample from data source with right classification already assigned. Such systems are used in feed forward or Multilayer Perception (MLP) models. These MLP has three distinctive characteristics [63]:

One or more layers of shrouded neurons that are not part of the info or yield layers of the system that enable the system to learn and take care of any mind boggling issues, The

non linearity reflected in the neuronal activity is differentiable and, The interconnection model of the system displays a high degree of connectivity.

These characteristics along with learning through training take care of troublesome and different issues. The blunder rectification learning algorithm trains the system based on the data yield samples and discovers mistake signal, which is the distinction of the yield calculated and the wanted yield and adjusts the synaptic weights of the neurons that is proportional to the result of the blunder signal and the information instance of the synaptic weight.

2.4.2 Unsupervised Machine Learning

Unsupervised learning algorithm recognizes concealed patterns in unlabelled data [1, 46, 64]. This unsupervised alludes to the ability to learn and organize information without giving a blunder signal to evaluate the potential arrangement. The lack of bearing for the learning algorithm in unsupervised learning can at some point be advantageous, since it lets the algorithm to think back for patterns that have not been already considered. The main characteristics of Self-Organizing Maps (SOM) are [2]:

It transforms an approaching signal pattern of arbitrary measurement into one or 2 dimensional map and perform this transformation adaptively, The system speaks to feed forward structure with a single computational layer comprising of neurons arranged in lines and sections, At each stage of representation, each info signal is kept in its legitimate connection, Neurons dealing with firmly related bits of information are near one another and they communicate through synaptic associations, and The computational layer is also called as competitive layer since the neurons in the layer rival each other to end up active. Consequently, this learning algorithm is called competitive algorithm.

2.5 Malicious code (Botnet)

"Bot" is an abbreviated subsidiary of 'robot'. It is feasible for an assailant to control a ton of bots over botnet utilizing one another. Utilizing a solitary order, a Botmaster can take the control of numerous contaminated PCs. With these qualities, an aggressor can start a mass assault on Internet client's effectively [12, 15, 37]. A botnet is a gathering of many tainted machines; alluded to as zombies, under a basic Command-and-Control base (C&C). An aggressor can control numerous PCs utilizing a wide assortment of methods and utilizing an assortment of conventions, for example, IRC, HTTP, and P2P and so on. Once traded of, bots are customized to associate with a focal area (normally an IRC server), where the Botmaster could log in and issue orders to the signed in bots. This component basically implies that the correspondence is free, as telecast is dealt with by the IRC station [32]. Davide et al. [22] detected botnet activity based on the traffic behavior analysis by applying the classification algorithm. As Bots interact over legal connection channels, which makes it difficult for signature-based IDS to detect without prior knowledge of the attack. Internet Relay Chat (IRC) is the most prevalent communication scheme among traditional botnets until the early 2000s. After infection, the bot will locate and connect with an IRC server. The bot masters use IRC command and control (C&C) to established channels and communicate and control the bots. The bot master will try to keep the bots under control as long as possible [35]. New bots were introduced around the mid-2000 that uses Peer-to-Peer (P2P) systems to frame the correspondences system for bots. In P2P plans, singular bots act as both client and server as the system is flexible in that when hubs are taken disconnected from the net, these services might be shut naturally, considering the system to keep on operating under the aggressor's control [35,39]. Notwithstanding, P2P botnets have their own confinements established, for the most part,

in the higher inactivity hidden the charge and control transmission and the effect of such an inertness on bots synchronization.

2.5.1 Botnet Life Cycle

A common botnet can be made up of and kept in five stages including: beginning disease, optional infusion, association, vindictive command and control, redesign and upkeep [65]. Amid the underlying contamination stage, the assailant examines an objective subnet for known vulnerabilities, and taints casualty machines through various abuse strategies. After beginning contamination, in optional infusion stage, the tainted hosts execute a script known as shell-code. The shell code brings the picture of the genuine bot paired from the particular area by means of FTP, HTTP, and P2P. The bot double introduces itself on the objective machine. Once the bot project is introduced, the casualty PC turns into a "Zombie" and runs the malicious code. The bot application begins automatically every time the zombie is rebooted [12, 39]. Figure 2.2 shows the execution or life cycle of botnet.

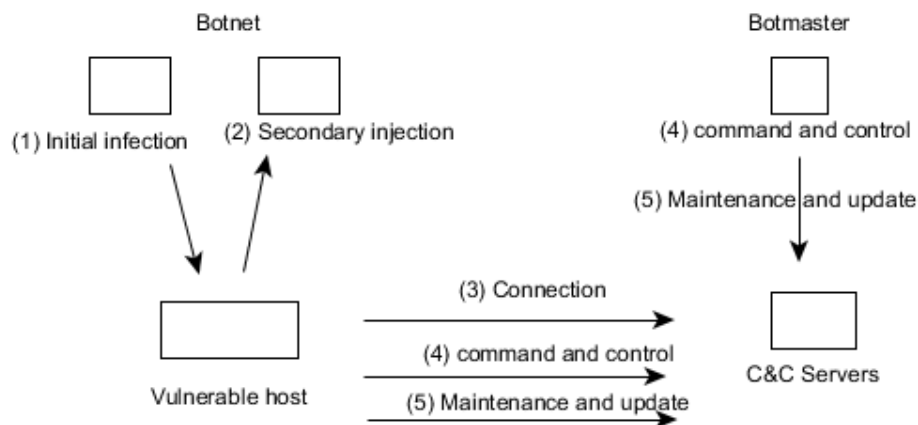


Figure 2.2: Botnet Lifecycle

In connection phase, the bot programs builds up a command and control (C&C) channel, and interface the zombie to the command and control(C&C) server. Upon the

foundation of C and C channel, the zombie turns out to be a piece of aggressor's botnet army. After connection phase, the real botnet command and control exercises will begin. The Botmaster utilizes the C and C channel to disperse commands sent by Botmaster [12, 29]. The C and C channel empowers the Botmaster to remotely control the activity of vast number of bots to direct different illegal exercises the purpose of the last phase, which is to keep up bots connections and upgrade. A bot controller upgrades their botnets from time to time, in other to enhance its attack patterns.

2.6 Indicators of Compromise (IOC)

Indicators of Compromise (IOCs) are forensic artifacts of an intrusion that can be identified on a host or network. The use of IOCs in detecting malwares or malicious code is one of the approaches that industries are applying to develop detection system, as the use of IOCs allows for a prompt detection and correlation of infected web pages or networks, even before they may be blocked by more traditional URLs blacklists [17].

Cyber Threat Intelligence (CTI) is defined as "evidence-based knowledge, including context, mechanisms, indicators, implications and actionable advice that can be used to inform decisions regarding the subject's response to an intrusion" [34]. IOC provides malicious code behavior knowledge that is essential to the organization, gaining visibility into the fast-evolving threat landscape, timely identify early signs of an attack and the adversary's strategies, tactics, and techniques, and effectively contain the attack with proper means. Due to the importance of Cyber Threat Intelligence, it has been collected and increasingly exchanged across organizations, often in the form of Indicators of Compromise (IOC), which are forensic artifacts of an intrusion such as virus signatures, IPs/domains of botnets, MD5 hashes of attack files, etc. Once collected, these IOCs can be

automatically transformed and fed into various defense mechanisms when they are formatted in accordance with a threat information sharing standard, such as OpenIOC, that enables characterization of sophisticated attacks like drive-by downloads [56]. The challenge, however, comes from the effective gathering of such information, which entails significant burdens for timely analyzing a large amount of data. Figure 2.3 shows example of IOC.

```
<definition>
  <Indicator id="427a68ab-6bd3-4f55-8be8-da463d173959" operator="OR">
    <IndicatorItem id="9b2784ef-4e0e-44d1-a6c3-416f4e28c323" condition="contains">
      <Context document="PortItem" search="PortItem/remoteIP" type="mir"/>
      <Content type="IP">137.175.4.132</Content>
    </IndicatorItem>
    <IndicatorItem id="637c784f-ee5f-4fdc-9109-54cb70b4550e" condition="is">
      <Context document="FileItem" search="FileItem/Md5sum" type="mir"/>
      <Content type="md5">079a440bee0f86d8a59ebc5c4b523a07</Content>
    </IndicatorItem>
    <IndicatorItem id="cdafa151-bbfb-4f47-8d3e-8c90a82b42b6" condition="contains">
      <Context document="PortItem" search="PortItem/remoteIP" type="mir"/>
      <Content type="IP">223.25.233.248</Content>
    </IndicatorItem>
    <Indicator id="a181f9e8-1c17-4a8b-a449-a7d4040f7a97" operator="AND">
      <IndicatorItem id="3c7d339e-57d8-45a0-b109-e989259ba040" condition="is">
        <Context document="FileItem" search="FileItem/FileName" type="mir"/>
        <Content type="string">movie.swf</Content>
      </IndicatorItem>
      <IndicatorItem id="413906a2-d3e3-4e99-aa79-bba242f5b1" condition="is">
        <Context document="FileItem" search="FileItem/SizeInBytes" type="mir"/>
        <Content type="int">214976</Content>
      </IndicatorItem>
    </Indicator>
  </Indicator>
</definition>
/ioc>
```

Figure 2.3: Example of IOC

Some research has been done in the area of IOC and there are some existing IOC which has been made available and can be use to enhance detection techniques [50]. Base on the finding or result of researchers, the existing IOCs are good intrusion detection features. This thesis will adapt and extract the existing IOC from the malware data collected. According to [50] "The analysis of malware capabilities from different datasets, including process monitoring, flow data, registry key changes, and network traffic will yield indicators of compromise". The researcher proposed the use of tools to collect IOC for the development defense mechanism for malware detection.

2.7 Related Works

Many research works have been done in the area of malware signature generation, malware clustering and classification or detection of malware attack using different techniques apart from using signature-based IDS.

2.7.1 Signature Generation using String Algorithm

A work was presented, an automatic generation of signature using honeycomb by [67]. Data collection and signature generation was done using honeypot system. Honeypot system uses Longest Common Substring (LCS) algorithm in its signature generation. Based on the work, the experimental result shows 649 TCP connections and 123 UDP connection of attack were made and by using the LCS algorithm, approximately 30 signatures were created, after which the signature was pass to Snort IDS for evaluation and Slammer worm was detected [67]. This approach used by the researchers is limited to known intrusion, as IDS need to have pre-defined or created signatures of the intrusion in the IDS signature database in other for a detection to occur. LCS algorithm as used to generate the signatures is a static algorithm that generates static signatures and in other to detect intrusion, the signature is match one by one to the incoming packets, if any match intrusion detection occurs and none match, intrusion traffic will be considered as legitimate.

Honeypot-based Signature Generation for Polymorphic Worms using a probabilistic approach was used by a researcher to capture suspicious traffic, analyze them and send them to the signature generator module for signature synthesis. The system extracts tokens from a suspicious sample and using a greedy probabilistic approach to generate a multi-sets String byte sequence as the signature of the polymorphic worm. The researchers used suffix array algorithm to induce sorting (SA-IS) for token extraction and to enhance the

speed of token extraction and memory efficiency. The disadvantage of this method is that it can still launch an attack in outbound limits. Instead of blocking all the outbound connections at the outbound threshold, some attacks are allowed periodically to pass through honeynet sensor to forward them to the IDS which will modify and disable the attacks [68]. This approach used by the researchers [68] are limited to known attack as the attack signatures must be in the intrusion system database in other for an intrusion alert to occur and it is prone to high false alarm rate.

Automatic SNORT Signatures Generation is implemented with a web based honeypot to generate Snort intrusion detection system signatures (Rules) for HTTP traffic [38]. The new signatures are integrated into the IDS signatures data base, and an evaluation is carried out to measure the effectiveness of the new signatures if able to identify and obstruct these attacks. The evaluation of web-based honeypot signature generation (WHASG) signatures was done by comparing after effect of original Snort signature and the new signatures. The work shows the signature (WHASG) has higher detection rate than the original Snort signatures. The outcome demonstrate that the new signatures had a low false negative with a percentage of 1 (one) and the existing one has 64. The false positive evaluation demonstrates that the newly generated signatures had higher false positive than the existing signatures with 5.53% for WHASG and 3.17% for existing signatures [38].

2.7.2 Malware Analysis and Detection

An approach to detect botnet activity based on traffic behavior analysis by classifying network traffic behavior utilizing machine learning was proposed. The work investigates on the attainability of detecting botnet activity without having seen an entire system stream by arranging conduct in light of time interims [68, 69]. The model allows detecting bot activity in both the command and control and attack phases based on the

observation of its network flow characteristics for specific time intervals. The concept of time intervals is to limit the duration to observe any particular flow before suspicions about the nature of the traffic may be raised [69].

With a decision tree classifier, the researchers were able to successfully detect botnet activity with high accuracy by simply observing small portions of a full network flow, allowing detecting and responding to botnet activity in real time. Experimental evaluation under various settings shows that their detector is able to achieve a true positive rate of over 90% and a false positive rate under 5%. They also demonstrated that the proposed detector is capable of identifying new threats without training on existing malicious datasets, by testing sample novel botnets. This approach has several benefits over traditional approaches based on group behavior and anomaly detection. First, it does not require significant malicious activity to occur before detection as it can recognize command and control signals, and it does not require the group behavior of several bots before it can be confident about making a decision [68, 70].

A work was presented on the use of two methods of malware analysis [71], namely:- Static analysis and Dynamic analysis. Static malware analysis is done without the execution of the malware, while dynamic analysis of malware is done with the execution of malware in a secure system [71]. When performing basic static malware analysis the first approach is the identification of the program that is alleged malware or not, also to detect packed/obfuscated technique used by malware, as well as finding malware creation time. Meanwhile, advanced malware static analysis methods is capable of providing more complete information about characteristics of malware, such as the information of malware to infect other programs, as well as modifying the registry and create new files and folders. Whereas on basic methods of malware dynamic analysis can discover Dynamic

Link Library (DLL) of malware, the process of malware inside the system, as well as the network connection performed by malware against the server.

Meanwhile, malware analysis with the advanced dynamic analysis method can provide information not previously found by other methods, the malware is able to turn off windows security systems such as firewalls, antivirus and system restore. The researchers stated *“the merging of the two methods of malware analysis that is static analysis and dynamic analysis is able to provide a more complete picture of the characteristics of malware TT.exe”* [93].

2.7.3 Signature Generation and Intrusion Detection using Machine learning

An Automated Signature Creator for a Signature-based Intrusion Detection System with Network attack Detection capabilities (PANCAKES). The proposed system utilizes Anomaly-based IDS (Bro IDS) to pre-identify malicious traffic coming from a network of virtual machines and/or from a data that is to be processed by the IDS. The Logs are the output of Bro IDS, which was applied on Pancakes (signature generator) in order to correlate the logs to the network traffic captured, which would help determine if a certain traffic that has passed through the network is a malicious traffic or not. All data coming from Bro IDS is stored in a database of log files. The log files contain the attacks file generated by Bro IDS, which will be applied to Pancakes for signature generation. The researcher also did some feature selection on the data collected by Bro IDS and applied machine learning algorithm such as: C4.5, Random Tree, Random Forest and Logistic Regression) to classify the dataset in order to determine False Alarm, correct and incorrect instances.

Based on the machine learning results, the analysis shows that Random Tree Algorithm is the most efficient algorithm because it has exhibited the highest percentage for Correctly Classified Instances. Through machine learning, the system was able to further determine the malicious instances within a capture file that is already preanalyzed by an Anomaly Based IDS. In addition to this, the system was able to predetermine instances that are classified to be network attacks [23]. Based on the researchers work, the proposed system is limited to Static or String signature generation as Pancakes is a static tool which uses Longest String Algorithm for Signature generation.

F-Sign, unique malware files signature extraction. The work [72] applied two approaches to examine malicious executables, such as: disassembly, using IDA-Pro, and the use of a dedicated state machine in order to obtain the set of functions comprising the executables. The findings suggest that the proposed strategy can be utilized for automatically generating signatures that are both particular and delicate. F-Sign is intended to produce basic, byte-string, signatures that can be utilized by network-based Intrusion detection system (NIDS) for separating malware continuously [72]. F-Sign was utilized as the programmed signature age (ASG) module of the eDare (early location, caution, and reaction) system, which offers "malware sifting as an administration". eDare is known for relieving the spread of both known and obscure malware in PC networks. eDare works by first checking network activity and sifting through known malware utilizing fast separating gadgets that are consistently updated with signatures produced by F-Sign. At the point when eDare recognizes another risk, F-Sign automatically creates a signature, and afterward, the sifting gadgets that are positioned on the network infrastructure are automatically updated [73].

Double-honey net system for sample worm collection and generation of signature from the collected sample. The sample data is collected from honeynet-2 after outbound connection is made from honeynet-1. Based on the research, is only malicious packets that make an outbound connection, because of the traffic it generate in the network trying to access resources. During this period, it makes many connections. After which machine learning (C4.5 algorithm) is applied to the collected data and signature generated and implemented into IDS system. The new signature was able to detect an unknown pattern and computational complexity [64].

An accurate signature generation system for Zero-day polymorphic worms is a novel Double-honeynet system, which is able to detect Zero-day polymorphic worms that have not been seen before. To generate signatures for polymorphic worms, we have two steps. The first step is the polymorphic worms sample collection which is done by the Double-honeynet system. The second step is the signature generation for the collected samples which is done by a string matching (Aho-Corasick Algorithm). The main goal for this system is to get accurate signatures for Zero-day polymorphic worm [65].

2.7.4 Clustering and Classification on malware data

Data mining technique is used to present the concept of attack data analysis and traffic classification on honeypot log file. All captured files are processed for feature selection and ML based traffic classification algorithm applied on the data. After the application of Navie-Bayes classification algorithm, the total correct classified instances is 4766 (66.1118%) and total incorrect classified instances is 2443 (33.8882%) from a total instance is 7209. This accuracy was based on the limited features of the traffic classification which leaves the scope for improvements by applying more and optimized

features [74]. This thesis looks at the clustering part of machine learning and malware analysis for feature selection. This research work performs an evaluation on the IOCRule which is a product of the malware data analysis, feature extraction, clustering and generation of behavior rule-based intrusion detection, beyond the work conducted by [75, 76].

A hybrid model integrates Anomaly based Intrusion detection technique with Signature-based Intrusion detection technique. The signature based IDS SNORT is used to generate alerts for anomaly data. Data mining techniques “K-means+ CART” is used to cascade K-means clustering and CART (Classification and Regression Trees) for classifying normal and abnormal activities on the alert generated. The evaluation shows total Number of Instances is 14957, while Correctly Classified Instances is 14870 (99.4183%) and Incorrectly Classified Instances 87 (0.5817%) using “K-means+ CART” and the result was compared to C4.5, which proofed that “K-means+ CART” is better than C4.5 in Time taken to build model and Correctly Classified Instances [44].

2.7.5 IOC as a Detection Techniques

Based on the review done by [61], one of the primary assets that data security individuals have on their attitude are Indicators of Compromise (IOCs), that permit the recognizable proof of conceivably vindictive movement on a system or network. Typically, IOCs are made of infection marks, IP addresses, URLs or domains and some others components that are not adequate to distinguish an interruption or noxious action on a computer system. The Windows event logs enlist distinctive exercises in Windows working systems that are important components in a scientific investigation preparation. In this study IOCs were generated utilizing Windows event logs for interruption discovery,

enhancing Incident Response (IR) and scientific investigation forms to accomplish a more proficient demonstrative computer system for IR [61]. To approve whether the IOCs generated are utilitarian or not, these were tried in a correlational instrument, Splunk. Splunk is a Security Information and Event Management Software that permits individuals to identify interruption events progressively. In an examination made on thirty computer systems over 60 minutes, 9996 events were enrolled, around 50% of these having a place with the classes of User Account Management, Logon and Logoff. These events were joined with others of higher event or lower event. This generated IOCs of basic or complex expressions, which diminished by additional than 80% the rate of false positives [61]. This thesis uses IOC technique for malware detection by developing an IOCRule, rule-based behavior for IDS, after a dynamic analysis and clustering of the Malware IOC extracted from the analyzed data.

"Indicators of Compromise TeslaCrypt Malware". The research work presents how to analyze live ransomware malware samples; within network and locally. Based on the research, analyzing live ransomware gives a unique three-dimensional perspective, visually locating crucial signatures and behaviors efficiently. The study also suggests that the analysis of malware capabilities from different datasets, including process monitoring, flow data, registry key changes, and network traffic yield indicators of compromise. These indicators are collected using open source tools such as Fiddler, Wireshark, and Snort. After the collection of malware IOC, it will be used to produce defensive mechanism against unwanted advanced adversary activity on a network. The researcher used a virtual appliance platform with simulated production Windows 8 O/S to collect indicators as the testing system is infected by malware and uses the tools to collect IOC to be used to secure enterprise systems [50]. The researchers [50] used tools to collect IOC from network

system and proposed the use of the collected IOC for defensive mechanism development. This study used honeypot system to collect malware data, analyze the data, and use java script code to extract IOC from the behavior analysis of the malware data. It then used OpenIOC tool to develop IOC logic and IOCRule for malware detection.

Benefit of using OpenIOC framework as a common syntax to describe the results of malware analysis. The researchers [76] stated, when investigating a malware specimen, it is important to do so in an isolated, “clean room” environment. The machines and network used in the analysis have to be isolated away from the production environment to prevent any possibility of malware outbreak. The behavior of the specimen should be analyzed in a clean installed machine that is not connected to external networks. By setting up such a existing approach, any changes made to the machines’ state can be attributed to the malware [76]. The study describes two types of analysis:

In static analysis, the specimen’s binary is examined without executing it. The first step in static analysis is file profiling which is done to obtain an initial assessment of the specimen’s functionalities. Information such as strings, library dependencies, meta data and anti-virus signatures can be extracted from the executable file.

In dynamic analysis, the behavior of the specimen is observed through its interaction with the host, as well as external system like web servers, IRC networks. When using OpenIOC framework to report the findings of the investigations, the mandatory and optional attributes can be expressed as AND and OR operators. OpenIOC is ideally suited for this purpose as the XML-based framework provides a flexible way of describing the complex semantics of a malware’s behavior. OpenIOC is focused on describing technical characteristics of a threat through an extensible XML schema. It has a comprehensive

vocabulary for describing low level attributes which can be easily translated into machine-understandable formats. These can then be used as input to configure various IT security monitoring and detection tools like anti-virus, IDS (Intrusion Detection System), IPS (Intrusion Prevention System), firewalls, OS (Operating System) security controls and policies [76].

iACE, an innovation solution for fully automated IOC extraction. The approach is based on the observation that the IOCs in technical articles are often described in a predictable way: being connected to a set of context terms (e.g., “download”) through stable grammatical relations. Leveraging on this observation, iACE is designed to automatically locate a putative IOC token (such as a zip file) and its context (such as “malware”, “download”) within the sentences in a technical article, and further analyze their relations through a novel application of graph mining techniques. Once the grammatical connection between the tokens is found to be in line with the way that the IOC is commonly presented, these tokens are extracted to generate an OpenIOC item that describes not only the indicator (such as a malicious zip file) but also its context (such as download from an external source) [77]. Running on 71,000 articles collected from 45 leading technical blogs, this novel approach demonstrates a remarkable performance, it generated 900K OpenIOC items with a precision of 95% and coverage over 90%, which is way beyond what the state-of-the-art NLP technique and industry IOC tool can achieve, at a speed of thousands of articles per hour. Further, by correlating the IOCs mined from the articles published over a 13 year span, this study sheds new light on the links across hundreds of seemingly unrelated attack instances, particularly their shared infrastructure resources, as well as the impacts of such open-source threat intelligence on security protection and evolution of attack strategies [77].

An automated approach to extract and validate IOCs for web applications, by analyzing the information collected through a high-interaction honeypot. Based on the experiments conducted by researchers it showed that the system is able to automatically generate web indicators of compromise that have been used by attackers for several months (and sometimes years) in the wild without being detected [78].

The first component of the system is a high-interaction honeypot that was used to observe the behavior of the attackers and collect the pages they modified or they uploaded into the vulnerable system. The role of this component is only to collect a large number of pages compromised by attackers. The second component of the system is in charge of pre-processing the collected data to automatically extract the URLs of the components remotely included in the attackers' files, and to store them along with some additional information in the database [78].

This approach used by the researchers has several advantages compared to traditional techniques used to detect malicious websites. First of all, not all the compromised web pages are malicious or harmful to the user. Some may be defaced to advertise product or services, and some may be part of legitimate programs to redirect users toward (less legitimate) online shopping websites. In any case, it is important to detect these pages to inform their owners and to alert the users on the fact that the content of the page has been compromised and cannot be trusted. Also in the case of more traditional drive-by-download pages, the use of IOCs allows for a prompt detection and correlation of infected pages, even before they are blocked by more traditional URLs blacklists [79, 80].

[81] Describe methods that a security team can use to posture their environment to detect advanced persistent threat activity based on adversary tactics, techniques, and

procedures. Active defense strategies typically focus on operators combing through packet capture and alert data. However, by moving to a strategy that centers on tripping up the adversary and pouncing on their trail, the network defender can supplement their current efforts by interdicting the intrusion kill chain directly [81]. In order to remain a step ahead of the Advanced Persistent Threat, it is important to remain conscious of the latest cyber defense strategies. To best defend a network it is important to adeptly manage all information resources associated with adversary activity. Universal cyber intelligence standards like OpenIOC, OVAL, and CybOX can facilitate easy threat information sharing between organizations. Repositories for this type of intelligence exist and should be leveraged where possible. Cataloging and sharing threat intelligence is a vital step along the path of combating cyber-attacks. Collating this data into a universal format further allows for pattern analysis of this data. Most importantly, it enables analysts to track trends, recommend additional defensive measures, and share information [81].

Table 2.2 reviews the related works and identifies the approaches, weakness of the works and compare it to this study.

Table 2.2: Related Works

Year	Author	Approach	Weakness
2011	Reshma & Chairag	30 signatures generated using longest common substring algorithm.	-Static signature. -Static algorithm. -Detect known attack.
2011	Shabtai et al.	Extraction of signatures from malware executables using IDA-Pro and dedicated state machine approach.	-Static signature. -Detect known attack. - Static algorithm.
2013	Hesham & Khalid	A web based honey pot to detect and generate signature.	-Static signature. -Static algorithm. -Detect known attack.
2013	Ocampo, Castillo & Gomez	Applied machine learning on generated signatures. Data retrieved from Bro IDS.	-Static signature. -Detect known attack
2013	Zhao et al.	Botnet activity detection based on traffic behavior analysis, which is done by classifying the behavior of network traffic utilizing machine learning.	-Used labeled dataset for evaluation..
2014	Sounak & Bimal	Signature generation using probabilistic approach.	-Static signature generation. -Detect known attack

Table 2.2 continued

2014	Rajesh & Tajinder.	Apply classification method on collected data to know incorrect and correct classified data.	Use the classification for comparison of correct and incorrect data.
2015	Jaina & Krunal	- Signature based use for data collection. Applied K-means+ classification and regression tress (CART)	- Data collection done using signature based IDS. Collect known attack data.
2015	Mohssen, Eisa & Neco	Used double-honeynet for sample worm collection and signature generation. Applied machine learning algorithm on the generated signatures.	-Static signature. -Strings algorithm
2015	Syarif, Yudi & Imam	Malware analysis using static and dynamic Approach.	- Static analysis. – No execution of the malware in its analysis environment.
2015	Mohssen, Eisa & Neco	Signature generation for zero-day attack using String matching (Aho-Corasick) algorithm.	-String-based (Static) signature.

Table 2.2 continued

2016	María, Tixteco, Pérez & Medina	Extract IOC from Windows events Logs.	-Focus on Known patterns.
2016	Liao et al.	fully automated IOC extraction with articles, Zip files etc. The IOC token are extracted to generate OpenIOC items for description.	-OpenIOC description and IOC extraction from paper articles.
2016	Catakoglu, Balduzzi M & Balzarotti	Automated approach to extract and validate IOCs for web applications. Used honeypot system to collect attacker's information, pre-process the collected data.	-No clustering. –No rule based for attack detection.
2017	Kelly	Malware analysis based on process monitoring, flow data, registry key changes, and network traffic. – IOC TeslaCrypt malware identification using Fiddler, Wireshark and Snort tools.	IOC collected can be used for known attack defensive mechanism against.

2.8 Weakness of Reviewed Work

Based on the reviewed works, the approach adapted by “Polymorphic Worms Detection Using Longest Common Substring” published by Mohssen, Eisa & Neco (2015) [68], is compared to this thesis proposed approach for authentication and validation purpose. Mohssen, Eisa & Neco (2015) used double honeypot for intrusion data collection and applied string algorithm for IDS signature generation. This thesis proposed work addresses the disadvantage of the existing approach as it overcome the problem of static approach of intrusion detection which focuses only on known type of intrusion attack.

2.9 Summary

This chapter reviewed several research works related to this thesis presented findings on how some techniques, and tools were to achieve a specific goal. This chapter helped to give significance to this thesis as related works were reviewed and several differences were found. Three key findings that can be concluded from this chapter are: (1) there are very few researches that directly tackled the issue of behavior approach in IDS detection, (2) there are researches that tackled Malware detection but in different forms or perspective such as using static approach, and (3) there are researchers that tackled intrusion detection using IOC, but from the perspective of using it to monitor windows log file or system files. These findings provide sufficient motivation for this research works.

CHAPTER 3

FRAMEWORK AND METHODS

3.1 Introduction

This chapter explains the design process, the steps taken, methods, and tools involved in the framework of this research which includes: general design of the research framework, design of the evaluation and experimental study and evaluation parameters.

3.2 Framework Design

This section discusses the impact of proposed solution on traditional or static approach to the intrusion-detection system. Malicious code type of attack is unpredictable and dynamic, so good studies and techniques are needed to measure and create advanced ways to detect and prevent the network from being compromised. This framework as presented in Figure 3.1 shows the steps and techniques to detect, collect data, analysis, evaluate and validate the research.

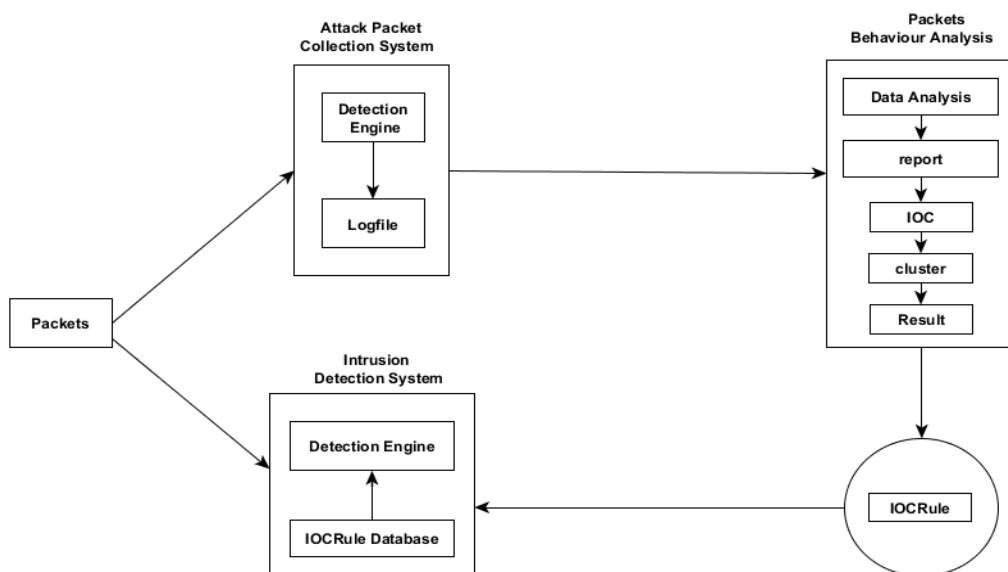


Figure 3.1: Research Framework

3.2.1 Intrusion Detection System (IDS)

Snort is an open-source Network Intrusion Detection System (NIDS) that has given researchers the privilege to modify and extend the system, and its functionality [43, 83]. The NIDS monitors networks and servers to detect intrusion activities [8, 21, 31]. Snort has the capacity to analyze real-time traffic and data flow in the network, as it is used to evaluate the performance of IOCRule over the traditional detection technique. Snort was selected among other types of IDS, as explained in Chapter 2.

Basically Snort consists of the following major components as shown in Figure 3.2 [82]. After the malware data collection, analysis, IOC extraction and IOCRule generation, the IOCRule is imported into the Snort IDS for the research evaluation.

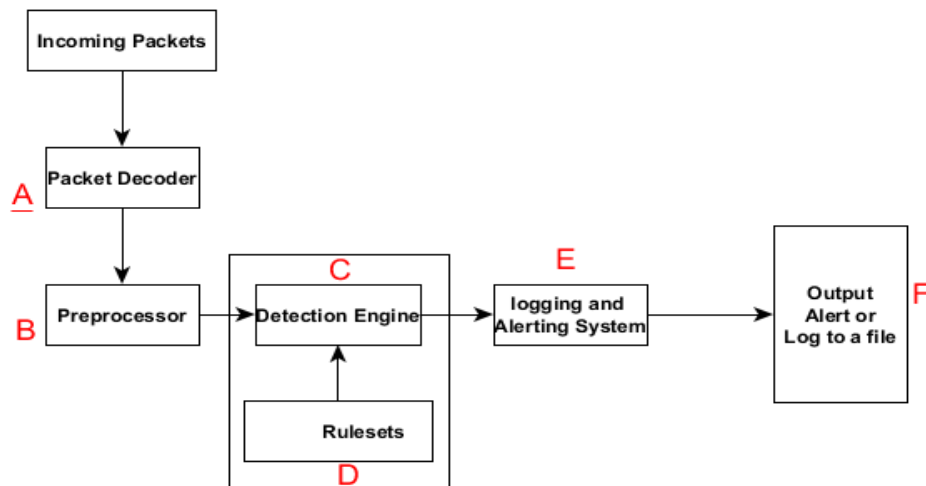


Figure 3.2: Snort System Architecture

Step A: The packet decoder collects packet from different network interfaces which might be Ethernet and SLIP and then send to detection engine. Figure 3.3 shows an alert that was captured by the signature stored in the database. This alert shows that someone is accessing facebook in the network. The snort packet decoder collects the incoming packets

and identified that it was facebook packets. Then, the next section process is followed and an alert is generated.

```

--== Initialization Complete ==--

--* Snort! <*-
Version 2.9.7.0 GRE (Build 149)
By Martin Roesch & The Snort Team: http://www.snort.org/contactteam
Copyright (C) 2014 Cisco and/or its affiliates. All rights reserved.
Copyright (C) 1998-2013 Sourcefire, Inc., et al.
Using libpcap version 1.5.3
Using PCRE version: 8.31 2012-07-06
Using ZLIB version: 1.2.8

Rules Engine: SF_SNORT_DETECTION_ENGINE Version 2.4 <Build 1>
Preprocessor Object: SF_DNP3 Version 1.1 <Build 1>
Preprocessor Object: SF_POP Version 1.0 <Build 1>
Preprocessor Object: SF_MODBUS Version 1.1 <Build 1>
Preprocessor Object: SF_GTP Version 1.1 <Build 1>
Preprocessor Object: SF_DNS Version 1.1 <Build 4>
Preprocessor Object: SF_SIP Version 1.1 <Build 1>
Preprocessor Object: SF_SSH Version 1.1 <Build 3>
Preprocessor Object: SF_IMAP Version 1.0 <Build 1>
Preprocessor Object: SF_DCERPC2 Version 1.0 <Build 3>
Preprocessor Object: SF_FTPTELNET Version 1.2 <Build 13>
Preprocessor Object: SF_SSLPP Version 1.1 <Build 4>
Preprocessor Object: SF_SMTP Version 1.1 <Build 9>
Preprocessor Object: SF_REPUTATION Version 1.1 <Build 1>
Preprocessor Object: SF_SOP Version 1.1 <Build 1>

commencing packet processing (pid=2888)
01/19-22:59:50.570748 ** [1:10003:0] kill content found ** [Priority: 0] (TCP) 192.168.26.131:39375 -> 202.71.99.194:80
01/19-23:01:10.820573 ** [1:10002:0] Ping from 192.168.26.130 ** [Priority: 0] (ICMP) 192.168.26.130 -> 192.168.26.131
01/19-23:01:11.826115 ** [1:10002:0] Ping from 192.168.26.130 ** [Priority: 0] (ICMP) 192.168.26.130 -> 192.168.26.131
01/19-23:01:12.829196 ** [1:10002:0] Ping from 192.168.26.130 ** [Priority: 0] (ICMP) 192.168.26.130 -> 192.168.26.131
01/19-23:02:07.767111 ** [1:100001:0] someone is accessing facebook ** [Priority: 0] (TCP) 192.168.26.130:32987 -> 31.13.67.1:443
01/19-23:05:34.379262 ** [1:100001:0] someone is accessing facebook ** [Priority: 0] (TCP) 192.168.26.131:37480 -> 31.13.79.246:443
01/19-23:05:34.415295 ** [1:100001:0] someone is accessing facebook ** [Priority: 0] (TCP) 192.168.26.131:59052 -> 31.13.67.1:443

```

Figure 3.3: Snort Detection Alert

Step B: The pre-processors work with snort to arrange the packet before detection engine apply some operation if the packet is corrupted. Basically it matches pattern of string, by sequence changing. Usually by adding some extra value, the intruder can fool the IDS but the pre-processor re-arranges the string to enable IDS to detect the string.

Step C: The detection engine identifies the intrusion activity in a packet with the help of snort rules and if found, appropriate rule action is applied else it drops the packet. The detection engine monitors the packets from the pre-processor and performs two way communications with the Rule set or Signature database, to check signature match of packets. If there are any matches, it generates an alert and log it.

Step D: This module stores the entire rule that snort IDS uses for detection purposes. Step A showed the alert from Facebook access and Figure 3.4 shows the rule that is used in

deriving the alert. The snort rule is a static rule, which matches packets by Strings. It can only detect an intrusion that matches the string of stored rule in the database. This type of rule tend to have high false negative as it is unable to detect undefined string in the database, but can only match one to one packets to the rules, based on the aim of this research to develop a dynamic rule (IOCRule), which is base on Malicious code behavior (IOC). The IOCRule then replaces Snort original signatures.

```
alert tcp any any - any any (content:"www.facebook.com";msd:"Someone is accessing facebook";sid:10002 ;)
alert tcp HOME NET any - EXTERNAL NET 80(msg:"kill content found";content:"kill";nocase;sid:1000002 ;)
alert icmp 192.43 any - any any (msg:"Ping found from 192.43";sid:10002 ;)
```

Figure 3.4: Example of Snort Rule

Step E: Logging and Alerting System work with what the detection engine finds in the packet and it might generate an alert or log activity. All log files are kept under /var/log/snort folder by default.

Step F: Output modules save output generated by the logging and alerting system.

3.2.2 Attack Packet Collection System

The attack packet collection module is the process of the malware data collection which is the training dataset needed for the research. T-Pot honeypot system is implemented in a network, and incoming traffic is monitored whereby the intrusion traffic analyzed, and data stored in the log file. Detection systems use the log file to record intrusion activities to enable administrators to study the intrusion information [84, 85, 86]. These systems are selected for this research work because of their availability as an open-source system, easy implementation and research community support and community intrusion data that can be retrieved using this system that contributes in giving this research a good attack data for analysis and evaluation. The first principal functionality of these

systems works through a Decision Engine that allows clients to filter incoming traffic based on multiple criteria, such as source IP or payload content, and a Control Engine that automatically limits outgoing traffic to prevent the system being compromised from external host's attacks [83, 87]. The second function of the system utilizes a Redirection Engine that straightforwardly diverts live network sessions (TCP or UDP) from one essential destination host to a secondary destination host utilizing a replay process. Honeypots are planned and conveyed to gather intrusion datasets that help in measuring and understanding network dangers [88, 89].

3.2.2.1 T-Pot Honeypot System

T-Pot is a honeypot system that is based on docker, which includes dockerized versions of the following honeypots:- conpot, cowire, dionaea, elasticpot, emobility, glastopf and honeytrap. Docker is a NAT device that has the ability to act as an SSH proxy between the attacker and the honeypot (Docker container in that case) and logs the attacker's activities. T -Pot uses suricata a Network Security Monitoring engine, and the ELK stacks to beautifully visualize all the events captured by T-Pot [73]. T-Pot is based on Ubuntu Server 14.04.4 LTS. The honeypot daemons, as well as other support components, are used to para-virtualized docker. Para-virtualization is an upgrade of virtualization technology in which a guest OS is recompiled preceding installation inside a virtual machine. This allows the running of multiple honeypot daemons on the same network interface without problems, and it makes the entire system as of very low maintenance [5, 73]. The encapsulation of honeypot daemons in docker provides a good isolation of the runtime environments and easy update mechanisms. This honeypot system comprises of different detection engine and types of honeypot such as a low, high interaction honeypot system.

In this research, T-Pot will be used to collect intrusion packets in the network. T-Pot as the honeypot type for this project is chosen from other types of honeypot such as:- Honeyd, Ghost-usb, Tiny Honeypot, Nephenthes, LaBrea, Kippo, KFSensor and Honeytrap; because of its availability and, easy installation. T-pot complies with the features of low and high interaction honeypot and it also allows users to view log intrusion attacks from other geographical zones, as it makes available collected intrusion data public. Figure 3.5 shows the architecture of T-Pot [90, 91].

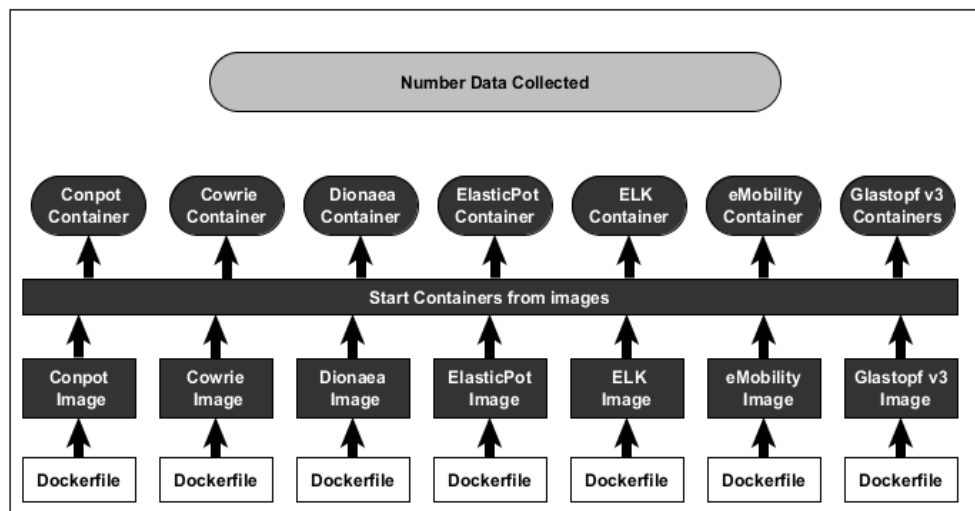


Figure 3.5: T-Pot Architecture to collect intrusion packets in the network

Based on the architecture design (Figure 3.5), T-Pot is made up of many other detection engines for data detection and collection. One of its best features is the selection of community dataset from the cloud database based on the need of the researcher. All data collected by T-Pot system are stored in the cloud, as this gives better security to a Local Network because the malicious data are not stored in the local network.

3.2.2.2 Architecture Details

All data in docker are volatile. Once a docker container crashes, all data created inside its environment are gone and a fresh instance is restarted. Henceforth, for a few data that should be persistent, such as config documents, a persistent storage/data/on the host keeping in mind the end goal is to make it accessible and persistent across container or when system restarts. Imperative log data is currently additionally put away outside the container *in/data/<container-name>* enabling simple access to logs from inside the host [92].

Basically, when the system is booted up, the following happens:

The host system start including important administrations (i.e. docker-engine), and all docker containers (honeypots, nms, elk)

3.2.3 Packets Behavior Analysis

This section elaborates each component of the Packet behavior analysis stage based on Figure 3.1. It explains how the malware data is analyzed and IOCRules generated for intrusion detection system

3.2.3.1 Data analysis

A sandbox framework is used to perform live data analysis on the attack data from the honeypot and extract behavior features of the malware data. Sandbox is a security analysis mechanism that is used for malware data analysis as well as analyzed the behavior of the data [8, 93]. This system allows malware data to be analyzed without interacting with a working network environment. It may be seen as a specific example of virtualization. The focus of sandboxing is to test unverified programs, which may contain malicious code, without allowing the malware to harm the host or network. Based on the

scope of this research, after the malware data is analyzed, the framework produces a report which contains the results of the analyzed data. This report has sufficient information needed for feature extraction, clustering and the creation of IOCRule.

3.2.3.2 IOC Clustering

Clustering is the process of data feature grouping. The focus of this research in clustering is to eliminate manual sorting of features, which can take months or years and this affect quick creation of defense mechanism for malware detection. To achieve this clustering purpose, Weka framework is used. Weka is a data mining system, that has the capacity to classify, cluster and preprocess data [94, 95, 96]. Hierarchical and K-means clustering algorithm is used for the clustering technique in Weka environment. After two clustering algorithm approach, the best algorithm for this research is selected. IOC is extracted from the report produced by the cuckoo sandbox, after which the IOC is inputted into Weka to be clustered to enhance the effectiveness of the IOCRule for malware detection.

3.2.3.2.1 Hierarchical

Hierarchical Clustering is an unsupervised learning method of cluster analysis, which seeks to build a hierarchy of clusters. It builds a cluster hierarchy, also known as a dendrogram [77][97]. Every cluster node contains child clusters; sibling clusters partition the points covered by their common parent. Hierarchical Formula is explains as follows:

Given a set of N items to be clustered, and an $N*N$ distance [98, 99].

Where,

N = Number of Instances

$N D$ = Distance between the instances

$K = \text{Number of clusters}$

Assign N item to K cluster, so that if we have N items, we now have K cluster, each containing just one item.

Let the distances (similarities) between the K clusters be the same as the distances (similarities) between the N items they contain.

Find the closest (most similar) pair of K clusters and merge them into a single cluster, so that now we have one cluster less.

Compute distances (similarities) between the new cluster and each of the old clusters.

Repeat steps 2 and 3 until all items are clustered into K number of clusters.

Advantages of Hierarchical:- No apriori information about the number of clusters required and easy to implement and gives best result in some cases, as it automatically assign number of clusters.

Disadvantages of Hierarchical:- Sometimes it is difficult to identify the correct number of clusters by the Dendrogram, Sensitivity to noise and outliers and Difficult handling different sized clusters and convex shapes.

3.2.3.4 K-Means

Clustering is an unsupervised technique which takes a different approach by grouping objects into meaningful sub classes so that individuals from the same cluster are entirely comparative and different as the individuals from the different cluster [69, 84]. The unsupervised detection algorithm clusters the unlabeled data instances together into clusters utilizing a basic separation based metric [62,100]. When data is clustered around the majority of the instances that show up in little clusters, they are labeled oddities. Since ordinary instances ought to frame vast clusters contrasted with interruptions and typical instances are subjectively different, they do not fall into the same clusters. In this research,

K-means will be used for the clustering algorithm. K-means is an iterative clustering algorithm in which items are processed among the set of clusters until they reach a desired set [99]. K-means formula is shown as follows [101, 102]:

$$J = \sum_{j=1}^k \sum_{i=1}^n ||x_i^{(j)} - c_j||^2$$

Where $||x_i^{(j)} - c_j||^2$ is a chosen distance measure between a data point “ $x_i^{(j)}$ ” and the cluster centre “ c_j ”, is an indicator of the distance of the n data points from their respective cluster centers. The algorithm is composed of the following steps:

Place K points into the space represented by the objects that are being clustered, points represent initial group centroids, Assign each object to the group that has the closest centroid, When all objects have been assigned, recalculate the positions of the K centroids, and Repeat Steps 2 and 3 until the centroids no longer move. This produces a separation of the objects into groups from which the metric to be minimized can be calculated.

Advantages of K-means are :- Fast, robust and easier to understand, Gives best result when data set are distinct or well separated from each other.

Disadvantages of K-means are: -The learning algorithm requires apriori specification of the number of cluster centers, the use of Exclusive Assignment - If there are two highly overlapping data then K-means will not be able to resolve that there are two clusters, Applicable only when mean is defined i.e. fails for categorical data, and unable to handle noisy data and outliers.

3.2.3.3 IOC-Writer

OpenIOC is an IOC framework used to create IOC logics and makes it easy to convert into other detection rules [56]. To formulate this research on IOCRule, the clustered results are used in the formation following the IOCRule procedures. Utilizing the OpenIOC framework, the IOCRule is presented in XML format and logical symbols (AND OR) to link or combine one IOC logical thought with another. Figure 3.8 shows the example of IOC created with OpenIOC.

```
<definition>
  <Indicator id="427a68ab-6bd3-4f55-8be8-da463d173959" operator="OR">
    <IndicatorItem id="9b2784ef-4e0e-44d1-a6c3-416f4e28c323" condition="contains">
      <Context document="PortItem" search="PortItem/remoteIP" type="mir"/>
      <Content type="IP">137.175.4.132</Content>
    </IndicatorItem>
    <IndicatorItem id="637c784f-ee5f-4fdc-9109-54cb70b4550e" condition="is">
      <Context document="FileItem" search="FileItem/Md5sum" type="mir"/>
      <Content type="md5">079a440bee0f86d8a59ebc5c4b523a07</Content>
    </IndicatorItem>
    <IndicatorItem id="cdafal51-bbfb-4f47-8d3e-8c90a82b42b6" condition="contains">
      <Context document="PortItem" search="PortItem/remoteIP" type="mir"/>
      <Content type="IP">223.25.233.248</Content>
    </IndicatorItem>
  </Indicator>
  <Indicator id="a181f9e8-1c17-4a8b-a449-a7d4040f7a97" operator="AND">
    <IndicatorItem id="3c7d339e-57d8-45a0-b109-e989259ba040" condition="is">
      <Context document="FileItem" search="FileItem/FileName" type="mir"/>
      <Content type="string">movie.swf</Content>
    </IndicatorItem>
    <IndicatorItem id="413906a2-d3e3-4e99-aa79-bba242fbe5b1" condition="is">
      <Context document="FileItem" search="FileItem/SizeInBytes" type="mir"/>
      <Content type="int">214976</Content>
    </IndicatorItem>
  </Indicator>
</Indicator>
</definition>
/ioc>
```

Figure 3.6: Indicators Examples

IOC is an artifact observed on a network or in an operating system that states with high confidence indicate an intrusion. Based on the IOCRule created using the OpenIOC framework, the Rule is then implemented into Snort IDS for malicious code detection and future attack detection.

3.3 Evaluation Parameters

This section explains the evaluation parameters of the research. These parameters are used to benchmark the evaluation study. Three parameters are selected and used to benchmark the existing approach and experimental studies. The selected parameters are:-

Number of Packets is the size of the evaluation dataset use in the performance evaluation of the research.

Detection rate is the number of attack detected by IDS. Intrusion Detection System performance can be measured by the number of detected attack. This can be calculated with the formula:

$$DR = A_d / A_p$$

Where,

DR = Detection Rate

A_d = Number of packet detected

A_p = Total Number of packet evaluated

The detection rate in percentage is derived by dividing the number of detected packets by total number of packets evaluated.

Detection time is the duration it took the detection tools or software to generate alert from the beginning of each dataset evaluation to the end, until the result is generated.

False negative shows the non-detection of intrusion packets for each detection techniques. In order to calculate this, false negative is the number of packets minus number of detected packets divided by number of packets multiple by 100.

$$FN = \frac{Np - Nd}{Np}$$

where,

FN = False Negative

$N_p = \text{Number of packets}$

$N_d = \text{Number of detected packets}$

In order to get the percentage of false negative, 100 is subtracted from the percentage of detection rate [103]. The above parameters are used to benchmark the experimental study. In order to verify the performance of the proposed solution of the research, the parameters are used to evaluate and the results compared to ascertain if the proposed result tally to problem statement. The parameters are used for thesis evaluation. Three parameters are used in this research, which are: False alarm (False Negative), detection rate and time. These parameters are evaluated, and the experimental study is carried out using CTU dataset.

3.3.1 Experimental Study

An experiment is a deliberate procedure carried out with the goal of verifying, defaming or establishing the validity of a hypothesis. This section explains the experimental study to be conducted in order to arrive at a conclusion, based on the data collected and analyzed. Performing experiment gives a stand in concluding a result. We focus on three experiments, which are used to test the hypothesis. The experiment is to evaluate the performance of IOCRule, as a proposed solution to Signature-based detection over malware.

3.3.1.1 Experiment 1: IOCRule with Different Clustering Algorithm

This experimental study is to identify the impact of different clustering algorithm on IOCRule, which is the result of IOC clustering. Table 3.3 shows the parameters and the aim of experiment 1 (one). The aim of this study is to identify the impact of clustering technique on the IOC. The following explains each parameter and its objective:-

Detection Rate: - By using different clustering techniques, there is a comparison of the detection rate achieved from the experimental study conducted using the IOCRule created from the outcome of clustered IOC extracted from analyzed malware data.

Detection Time: - The aim of evaluating with this parameters is to measure the detection time of the IOCRule created from the clustering technique.

False Negative Rate: - One of the aims of this study is to have a very low false alert. Each clustering technique (K-means and Hierarchical) result is compared over one another, to determine which produces low false negative.

Table 3.1: IOCRule with different Clustering Algorithm

Parameters	Aim
Detection Rate	To know the detection rate of IOCRule from different clustering techniques
Detection time	To know the detection time of IOCRule from different clustering techniques
False Negative Rate	To know the false negative rate of IOCRule from different clustering techniques

3.3.1.2 Experiment 2: Vary Number of IOCRule

Table 3.3 shows the parameters and the aim of Experiment 2 (two). The following explains each parameters and its objective:-

Detection Rate: - This is to identify the impact of the number of IOCRule to the detection rate. It shows if more or less number of IOCRule is better for intrusion detection.

Detection Time: - This is to show the detection time of varying number of IOCRule. The aim is to know the number of IOCRule that gives less detection time and with good detection rate.

False Negative Rate:- This signifies the impact of detection rate, when there is high detection rate, there will be less false negative.

Table 3.2: Vary Number of IOCRule

Parameters	Aim
Detection Rate	To compare the detection rate for different number of IOCRule
Detection time	To compare the detection time for different number of IOCRule
False Negative Rate	To compare the false negative rate for different number of IOCRule

3.3.1.3 Experiment 3: IOCRule with and without Clustering Algorithm

Table 3.4 shows the parameters and the objective of experiment 3 (three). The following explains each parameter and its objective:-

Detection Rate: - The aim of this study is to compare and identify the detection rate of IOCRule created from clustered IOC features and created without clustering technique. This helps to identify the impact of clustering on IOCRule.

Detection Time: - This is to show if the procedure used in creating the IOCRule has impact in the detection time. The aim is to compare the detection time of IOCRule created with clustering technique and without clustering technique.

False Negative Rate: - The outcome of this study is impacted by the detection rate. High detection rate reflect to low false negative.

3.4 Summary

This chapter discussed the techniques and tools that is used in this thesis. Identification of research steps are much important as it will guide in the research execution and better results. In order to proceed to the next stage of this thesis, I have discussed the tools and steps with the process of IOCRule create and implementation into IDS system for Malcode detection and experimental study to be carried out.

CHAPTER 4

RESULTS AND ANALYSIS

4.1 Introduction

The objective of this chapter is to conduct the experimental study explained in previous chapter and analyze the results. It shows the performance evaluation of proposed framework and existing framework.

4.2 Experimental Platform System Architecture

Hardware and software played an important role in this experimental environment setup, from the aspect of data collection, data analysis, IOC extraction, IOC clustering, IOC & IOCRule generation, until the results presentation. The empirical platform used includes the hardware and software requirements. Figure 4.1 shows the platform and its relationship to each other.

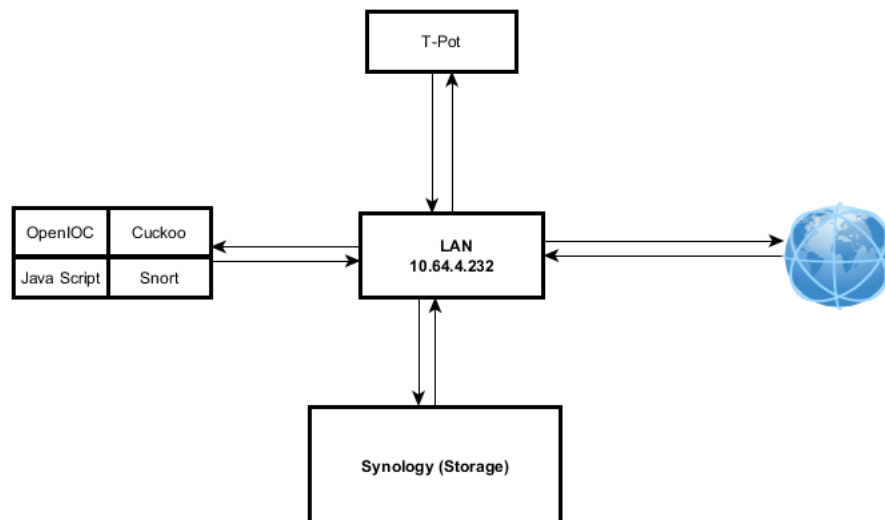


Figure 4.1: Experimental Platform System Architecture

T-Pot monitors live internet connection and collect intrusion or Malware data.

Synology stores the malware data collected by T-Pot.

Cuckoo is used to analyze the malware data, to do so. It obtains the malware data from the Synology for analysis.

Java Script is used to extract IOC from the analyzed malware data.

OpenIOC is used to write IOC logic, and create IOCRule.

Snort is the IDS, where the IOCRule created is implemented for evaluation and experimental study.

4.2.1 Hardware

Acer Aspire 4930 This is the system which house the T-Pot honeypot system and Sophos Firewall in a VMware (Virtual Machine).

Processor: Intel Core 2 Duo processor P7350

Ram: 8 GB

System Type: 64-bit Operating System, x64-based processor

Harddisk: 500GB

Dell Precision T3600 This Desktop system is where the Ubuntu Operating system, Snort IDS and cuckoo malware behavior analysis framework is installed for evaluation and experimental study.

Processor: Intel Xeon(R) CPU E5-1620 0

Ram: 16 GB

System Type: 64-bit Operating System, x64-based processor

Harddisk: 1TB

Synology: This is the system that all data collected and analyzed are stored. The Cuckoo analysis system fetches it analyzing malware data from the synology, as this will enhance the processing power of Dell Precision system.

Processor: Intel Atom C2538 x86_64

Ram: 8 GB

Harddisk 4TB

4.3 Formulated Hypothesis

A research hypothesis is a statement of expectation or prediction is tested by research. A research question flows from the topic that has been considered in this thesis. The research question, when stated in one sentence, is a Research Hypothesis. In the hypothesis writing, we predict the relationship between variables (Proposed techniques). The research questions formed are:

1. Is Dynamic approach better than static approach for malware data collection and analysis?
2. Is Dynamic analysis performance better than Static analysis for malware detection?
3. Is behavior approach of malware detection using IDS better than static approach of malware detection using IDS?
4. Is IOCRule detection technique which is derived from a behavior analysis method better than signature from static analysis?

With the hypothesis, the experimental study is conducted to evaluate the parameters used as a benchmark between proposed solution and existing approach study.

4.4 Comparison of Existing Work and Proposed Approach

The purpose of this comparison is to validate and point out the advantage of the proposed approach over existing solution as discussed in chapter 2, section 2.8. In this research, the comparison is used to measure the degree and quality of change from the

existing detection approach to the proposed detection approach. Figure 4.2 shows the steps for the comparison evaluation.

This section explains the evaluation of the comparison study using two different datasets (CTU and DARPA Dataset). CTU dataset used in this thesis is the 2016/2017 network intrusion dataset. The CTU-13 is a frequent dataset of botnet traffic that was captured in the CTU University, Czech Republic, on a year to year basis [104]. The goal of the dataset was to have a large capture of real botnet/malware traffic mixed. The CTU-13 dataset comprises of thirteen captures (called scenarios) of various botnet samples. On each scenario, an execution of a specific malware, which utilized several protocols and performed diverse actions, was completed. Each scenario was captured in a pcap record that contains all the packets of traffic (IRC, SPAM, Click Fraud, Port Scan, DDoS, FastFlux, HTTP/S) [98]. DARPA is a 1999 network intrusion dataset [57]. In the DARPA IDS evaluation dataset, all the network traffic including the entire payload of every packet was recorded in tcpdump format and provided for evaluation. The datasets are in the form of sniffed network traffic, Solaris BSM review information, Windows NT review information. The datasets comprised of a blend of genuine and mimicked machines; foundation traffic was misleadingly produced by the genuine and reenacted machines while the assaults were done against the genuine machines. The datasets comprised of long stretches of intrusion datasets, which comprised of typical traffic and comprises of marked assaults. This dataset was embraced in this study because of the sorts of intrusion traffic the dataset was made of, for example, Botnet, DDoS, phishing, rootkit. To perform this evaluation study, Snort IDS is configured and setup to analyze incoming and outgoing traffic in the setup network environment for this research. Snort reads every traffic and records numbers of packets that passed through the detection or analysis engine. Each of

the datasets is injected into the snort IDS and run on the eth1 (Ethernet interface). In this study, Snort uses the eth1 interface as its packet receiving interface packets go in and out through the interface.

Refer to Table 4.1 for the evaluation result. The following subsections explain the steps for the evaluation as shown in Figure 4.2.

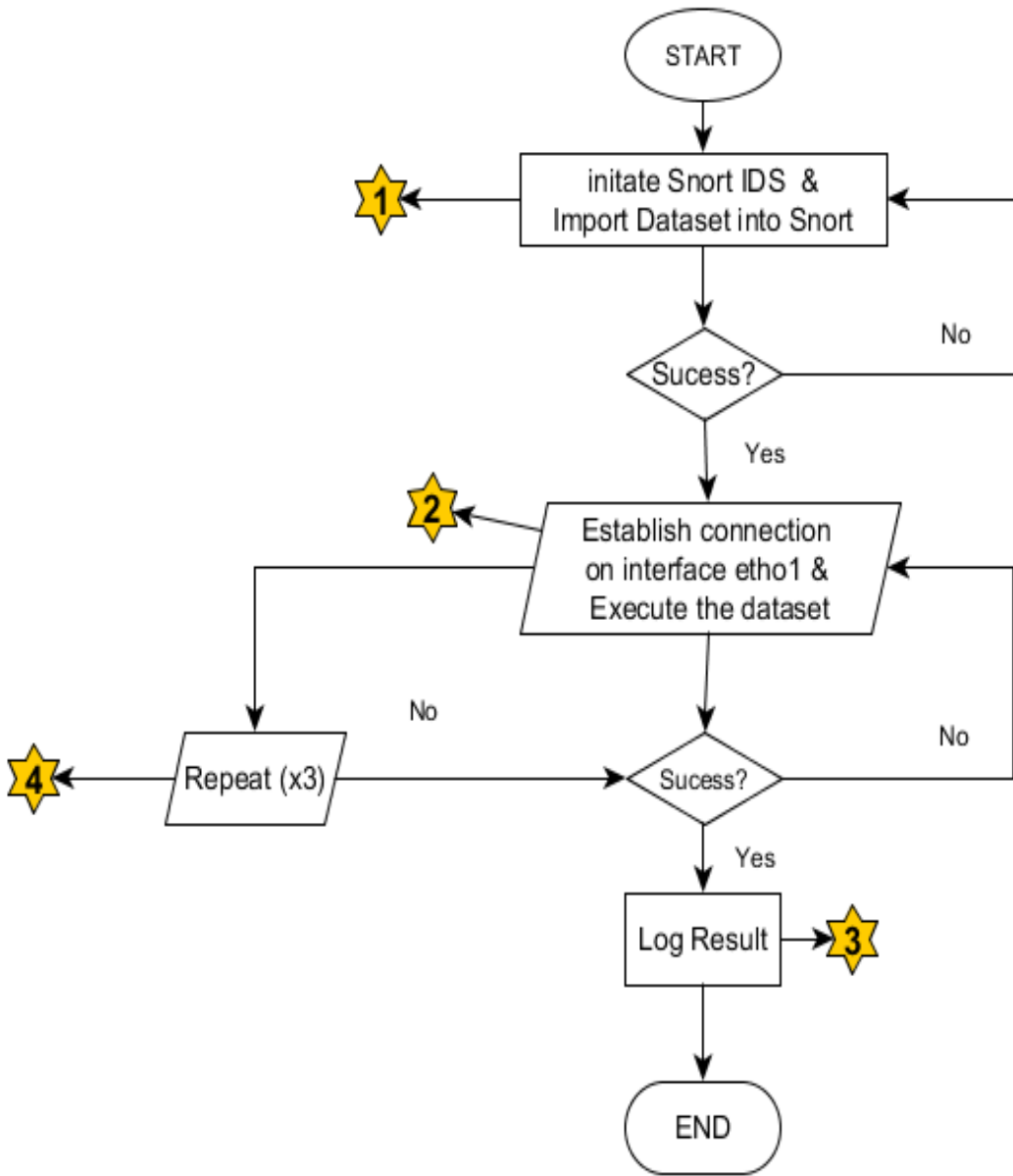


Figure 4.2: Comparison Evaluation Flow

The steps involved are as following:

Step 1: Initiate Snort IDS Process and Import Datasets- This stage is the starting up of Snort IDS and ready to receive instruction from the Linux terminal. After the start up Snort, the Datasets are imported into a directory in Snort.

Step 2: Establish Connection and Execute the Datasets- In order for snort to analyze and detect incoming traffic or packets, connection at the Ethernet interface needs to be established. After the connection is established, the Dataset is executed in the stored directory. The execution of the dataset activates the dataset to start running as an incoming traffic through the Ethernet interface and snort starts the analyzing and detection processes. To initiate the evaluation process in Snort, the following command is used: *Snortpcap dir = etcsnort folder where the Datasets is stored" C snort:conf*

Step 3: Log Result- The detection result is logged into the intrusion alert logfile database.

Step 4: Repeat the Process- Each dataset evaluation was done up to 3 times to arrive at a conclusion based on the result achieved.

4.4.1 Existing Approach Evaluation with DARPA Dataset

Figure 4.3 shows the evaluation result of 1999 DARPA datasets on traditional IDS (Signature-based Snort IDS), as it shows the performance of signature-based IDS. An IDS generates an alert when the signature in its database matches the intrusion packet.

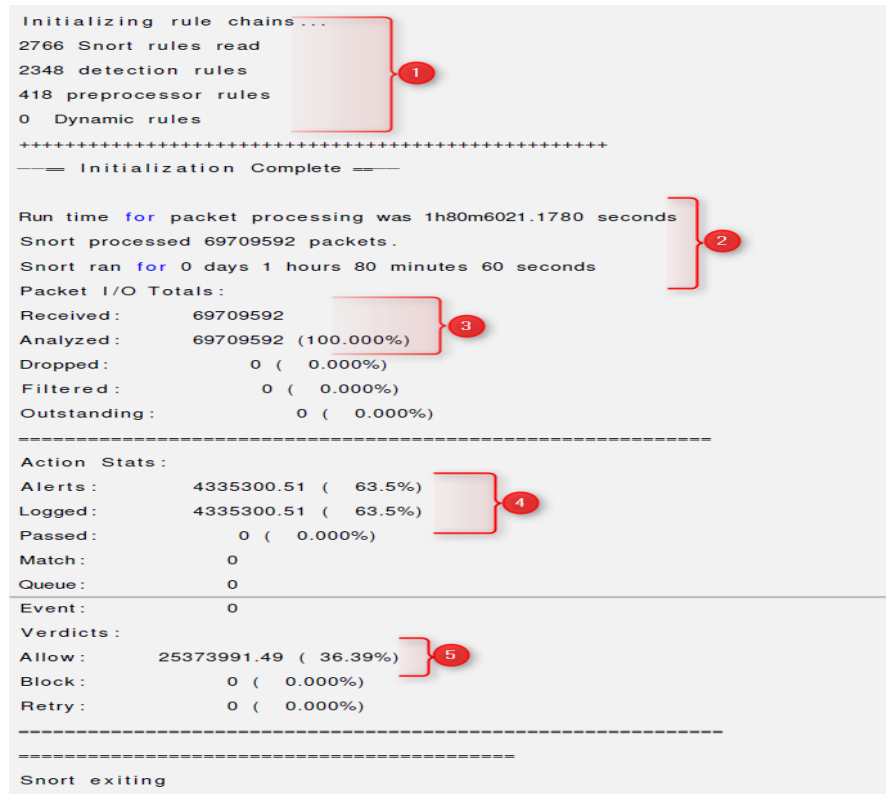


Figure 4.3: Evaluation result of DARPA Dataset

The steps involved are as following:

Step 1: The snorts first initialized the rules in the database and used them to detect intrusion from incoming traffic.

Step 2: This showed the run time for snort to process the DARPA datasets, and it showed total processed packets to be 69,709,592 and it took 1 hour 80 minutes and 60 seconds, which is referred to as the Detection Time.

Step 3: This showed the total packets received by the detection engine and we can notice, it received the total packets processed in label 2. After it received the packets, it proceeded to analyze them.

Step 4: Presented the percentage of the alert generated and logged from the analyzed packets, which is 63.5% from 100% of the packets, referred to as Detection Rate.

Step 5: This label presents the number of intrusion packets allowed to pass the detection engine without the rules generating an alert, which is referred to as False Negative Rate with the value 36.39%.

4.4.2 Existing Approach Evaluation with CTU Dataset

Figure 4.4 shows the evaluation result of CTU dataset on traditional IDS (Signature-based Snort IDS), as it shows the performance of signature-based IDS. Signature-based IDS generates an alert when the signature in its database matches the intrusion packet. CTU is a 2016/2017 malware dataset, and the result showed that signature-based IDS was unable to detect much percentage of the intrusion as it is only effective when there is a match in the database.



Figure 4.4: Evaluation result of CTU Dataset

Step 1: The snorts first initialized the rules in the database and used them to detect intrusion from incoming traffic.

Step 2: This showed the run time for snort to process the DARPA dataset, and it showed total processed packets to be 74,220,759 and it took 1 hour 43 minutes and 40 seconds, which was referred to as the Detection Time.

Step 3: This showed the total packets received by the detection engine and we can notice, it received the total packets processed in Label 2. After it received the packets, it proceeded to analyze them.

Step 4: Presented the percentage of the alert generated and logged from the analyzed packets, which is 1.184% from 100% of the packets, referred to as Detection Rate.

Step 5: This label presented the number of intrusion packets allowed to pass the detection engine without the rules generating an alert, which was referred to as False Negative Rate with the value 98.816%.

4.4.3 Existing Approach Study Discussion

Table 4.1 shows the existing approach evaluation result of this research based on CTU and DARPA Dataset [20]. The result presented in Table 4.1 shows the detection rate of signature IDS using 2016/2017 CTU malware dataset to be at 1.184%, 1 hour 43 minutes and 40 seconds detection time, and a false negative of 98.816%. It also shows the detection rate of Signature IDS using 1999 DARPA Dataset to be 63.5%, 1 hour 80 minutes 60 seconds Detection Time, False Negative 36.39%. The existing approach study has 2766 signatures stored in the IDS database for the evaluation study.

Table 4.1: Existing Approach Evaluation

Parameters	CTU	Darpa
No of packets	74220759	69709592
Detection Rate	890811(1.184%)	44335300.51(63.5%)
detection time	1h43m40s	1h80m60s
False Negative	73329882(98.816%)	25373991.49(36.39%)

Base on the evaluation result, DARPA dataset produced a better result than CTU dataset. The reason behind this is that DARPA is an old dataset and many signatures have been created for such intrusion attack, as this gave the Signature IDS better performance compared to CTU dataset, which is 2016/2017 malware dataset. The result also showed that signature-based IDS is unable to detect a high percentage of Unknown malware, as its signatures were not stored in the IDS database, and its detection approach is weak, as it used signature detection techniques, which was a static approach to detection.

4.5 Proposed Framework Evaluation

Figure 4.5 refers to the overall flow of the research framework. It explains the steps to be taken to achieve the objective of this study.

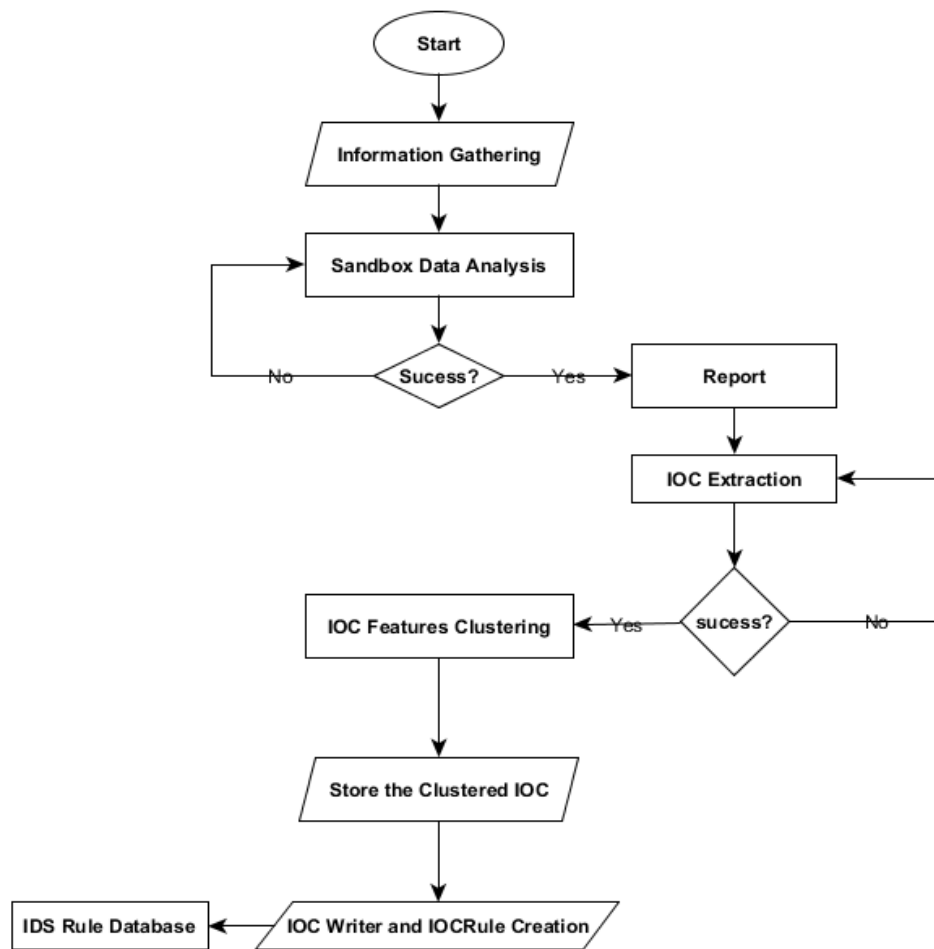


Figure 4.5: Evaluation Framework Flowchart

This framework is explained in details in the following subsections:

4.5.1 Information Gathering

Based on the research framework, the information gathering can be referred as data collection process. Many steps and techniques were implemented and adopted during data collection such as accessing malicious sites, downloading of malware files and executing malware files in the network environment for the data collection system (Honeypot) to record the intrusion. This process was to direct malware traffic into the honeypot. Some of the malware downloaded and executed include Rootkit, Zombies, Scareware, Virus Hoax,

Trap Door, and botnet. These malicious codes were downloaded from the site listed by Quttera [74].

Based on the data collected by the honeypot, a total number of 47462 malware were extracted from the T-Pot honeypot system in a period of one month and from different network environment through the wireless network and wired network. T-Pot system provided logfile monitor system which rechecked the logfile to filter out normal traffic that was recorded as an attack. Figure 4.6 refers to the sample of training dataset collected.

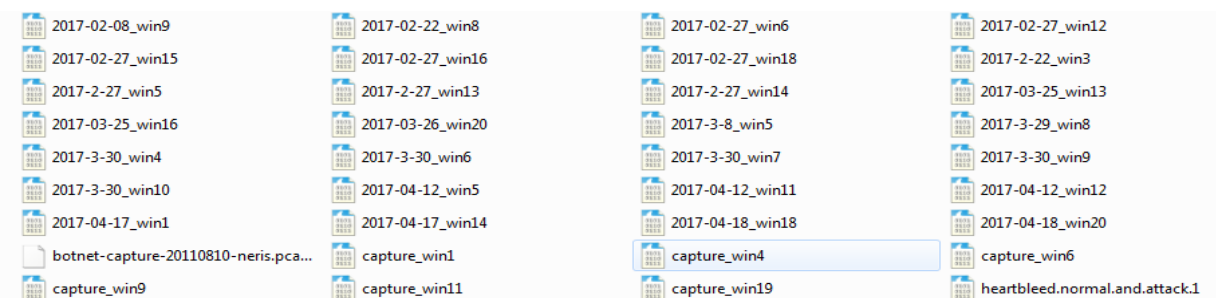


Figure 4.6: Malware Data

4.5.2 Sandbox Data Analysis

Based on the intrusion information gathered by T-Pot honey-pot, the training dataset was analyzed using the cuckoo sandbox framework, focusing on the behavior of intrusion data and network activities. The analysis period took a minimum of two weeks and three days. The analysis framework fetched the malware data from cloud storage platform (Network Synology) and applied its analysis techniques.

The link (<http://QuickConnect:toI0296905000001>) specifying the location of the malware data was specified and configured into the cuckoo system for direct fetching of the data for analysis. These methods reduced the system resource usage during the analysis

and rendered fast and better performance for quick delivery of analysis results. Figure 4.7 and Appendix B showed the result of the malware analysis.



```

{
  "machine": {
    "status": "completed",
    "name": "cuckoo1",
    "label": "cuckoo1",
    "manager": "VirtualBox",
    "started_on": "2016-12-07 10:52:57",
    "shutdown_on": "2016-12-07 14:14:02"
  },
  "target": {
    "category": "file",
    "file": {
      "yara": [],
      "sha1": "915934b43d63dc4040af3ea1ee6c80913288ff3b",
      "name": "rbot.exe",
      "type": "PE32 executable (GUI) Intel 80386, for MS Windows",
      "sha256": "dcf50510efec16ff10c5aed91c8e386aba114e63842ca16ea40cac776c60816",
      "urls": [
        "http://www.vapor.com/"
      ],
      "crc32": "B28D1B52",
      "path": "/home/rene/cuckoo/storage/binaries/dcf50510efec16ff10c5aed91c8e386aba",
      "ssdeep": "3072:X+ukBM5XiV/F/PY92m5fCHM9pvkSJQLqZNTAoEoY6PEZcvLKS4x/Q/HSL6:X+BI",
      "size": 225792,
      "sha512": "12712473c6e386c92b64df45c5570db673c6c435a50a9e93e09636264c877a177ff",
      "md5": "2467b3c8b259cecd6ce2d5c31009df10"
    }
  }
}

```

Figure 4.7: Data Analysis Result

1. This showed the information of the machine and the analysis status of the malware data using cuckoo sandbox, which can be identified as it completed the analysis.
2. Show the information of the analyzed file (rbot.exe). The information retrieved from the analyzed result, are used for the clustering based on the extracted features.

4.5.3 IOC Extraction

After the analysis, six (6) Indicators of Compromise (IOC) were extracted from the analyzed result. The extracted IOCs described the behavior of malicious activities. Indicators of Compromise (IOC) can be identified as anything that portray, the behavior or that can be a behavior (feature) of an intrusion. The extracted IOC in Table 4.2 were

clustered and used for NIDS IOCRule generation. Due to the focus of this research, six (6) IOCs were selected out of a number of possible IOCs that can be extracted depending on the focus of the researcher and the purpose of the research. The selection was done using the guide of existing IOCs and previous research [97]. IOCs are features of an item (malware) as discussed in chapter 2. Indicators of Compromise (IOC) can be anything, for example huge amount of traffic in a short period of time, file structure change (windows registry or in memory). Figure 4.8 shows the extracted IOC (features), and Appendix B shows the Java Programming code used for the extraction.

The IOC extracted are:-

DNS is selected as an IOC as malware is increasingly using a variety of domain techniques to take down efforts and make their command-and-control servers harder to locate. DNS traffic becomes a good indicator of compromise as indicated by security experts, as monitoring the network for strange DNS behavior can help pinpoint intrusion attack. DNS is inevitable when it comes to communicating across the network, as malware are unable to communicate without an active DNS.

Classtype shows the malware family of each analyzed file and the connection type of the malware. The classtype represented the malware family, as every malware belonged to a set of family, which portrayed a kind of behavior or similarity in their way of attack or invading a victim.

Sha1 & 3 were selected as IOC on the grounds that malware cannot be executed without digital certificates. Malware creators can profit by taking and marking their dangers with numerous digital certificates. As we exhibited Suckfly, one digital authentication can help make things less demanding for attackers. With two digital certificates, attackers stand far and away, getting superior shot of fulfilling their objectives.

The assaults with Carberp, likewise, indicated a move towards utilizing digital certificates with SHA2. SHA 1 and 2 were extracted from the malware dataset collected from honeypot to manufacture an IOCRule.

MD5 was selected as an IOC because it was designed to be used as a cryptographic hash function and malware used it to calculate cryptographic checksum of the file or target system or network.

Ssdeep helps to determine whether a suspected piece of malware is similar to a known malware sample. It also allowed analysts to compare a program on disk. Ssdeep was derived from a technique used to detect spam in email messages.

Reference was one of the ways malware writers established persistence within an infected host, through registry changes.

DNS	Classtype	Sha1	md5	Ssdeep	Reference
piwxvumpypptp.com	Trojan.Win32.Agent.dzdtfa	7351ae5ad8856ff3f6721c79799a50bcc73340af	2467b3c8b259cecd6ce2d5c31009df10	3072:X+ukBM5XiV/F/PY92m5fCHM9pvkSJQLqZNtAoEoY6PEZcvLKS4x/Q/HSL6:X+BBMRiVV8fn8Sy2ZAL/6PBLKS4ESL6	http://www.amazon.com/Jane-Greer/e/B001IXU0G2/ref=sr_ntt_srch_lnk_1?qid=1324346250
f02783mat0i5r1t.cc	RiskWare[Downloader].not-a-virus	64d8904a67be781214dde138fe882309a44619c0	fa02f2bdb89c20f0f162998984320c01	12288:WeDwOTE6SWJuFZYv8tYxi0SG+bBxKP8fzdrCAg2JGfwIsY8mxOXx:p9TE6zEFesuxA1b6P8bIAg2JGSmxOB	http://www.forensickb.com/2013/03/file-entropy-explained.html
piwxvumpypptp.net	Gen:Variant.Symmi.56366	7351ae5ad8856ff3f6721c79799a50bcc73340af	f589827c4cf94662544066b80bfd6ab	7zJ3+I/mKyHOeNvVLOUpSr	http://www.womenshealthmag.com/sex-and-relationships/what-is-the-sexiest-age
www.reimageplus.com	W32.Cleanspy.C.Trojan	5c2ba68d78471ff02adcdab12b2f82db8efe2104	bf08e6b02e00d2bc6dd493e93e69872f	768:3uKuCkdC2D5z4oWV0OCAB4Rld8Inv47erVe6ARm6v+kM8yR+4V:3Tjkdjl8WK4RP84vkelzvIM8yR+2	http://www.forensickb.com/2013/03/file-entropy-explained.html

Figure 4.8: Extracted IOC

4.5.4 IOC Clustering

Clustering techniques help to group related features in classes. Two clustering algorithm is used in this research, in other to draw a conclusion of which algorithm performs best with this research dataset. Kmeans and hierarchical clustering is used as the algorithm for effectiveness of this research. A total of 284771 IOC features were extracted

from 47462 malicious code analyzed by cuckoo sandbox. The clustering algorithm is applied to the extracted features, to group them into clusters to enable easy creation of IOCRule, as this helps to overcome the problem of Rules duplication. Each clustering algorithm is used on extracted IOCs, after which IOCRule is created from the clustered result. To perform clustering technique in this research Weka framework is used. Weka is a data mining system, which has the ability to perform clustering on a set of data.

Appendix B, Figure 1 shows weka interface which presents the number of instances in the dataset, number of Attributes and graphical visualization of all the instances of each attribute. Some elements like Number of samples, number of clusters are defined as they have an impact in the clustering result. The elements are explained as follows [17]:

Number of Samples (Instances):- This is the total number of features for the Six (6) IOC extracted. In each 47,462 analyzed result, 6 IOC features were extracted and the result were referred to as the number of sample (Instances).

Number of Clusters: - This is the number of group classes; the clustering algorithm produced after clustering process

4.5.4.1 Hierarchical Clustering

Hierarchical clustering (HC) was applied on nominal dataset using weka framework. After the clustering analysis, a total of 40 clusters were achieved using Agglomerative method, of Hierarchical clustering. Agglomerative also called bottom up is one of the method used in hierarchical clustering, this method start with 1one point (singleton), Recursively add two or more appropriate clusters to it Stop when k number of clusters is achieved. As discussed in Chapter three (3), it builds a hierarchy structure that every parent has a child. The uniqueness of this clustering algorithm is that it automatically updates its

clusters until it have a good clusters position. After the clustering analysis being executed, 40 clusters were selected. Hierarchical clustering can be represented by a dendrogram. Cutting a dendrogram at a certain level gives a set of clusters. Cutting at another level gives another set of clusters. The cluster selection is subjective depending on the researcher's needs and analysis. Figure 4.9 shows the dendrogram structure of the clustered result of this research. Refer to Appendix B for the clustering result.

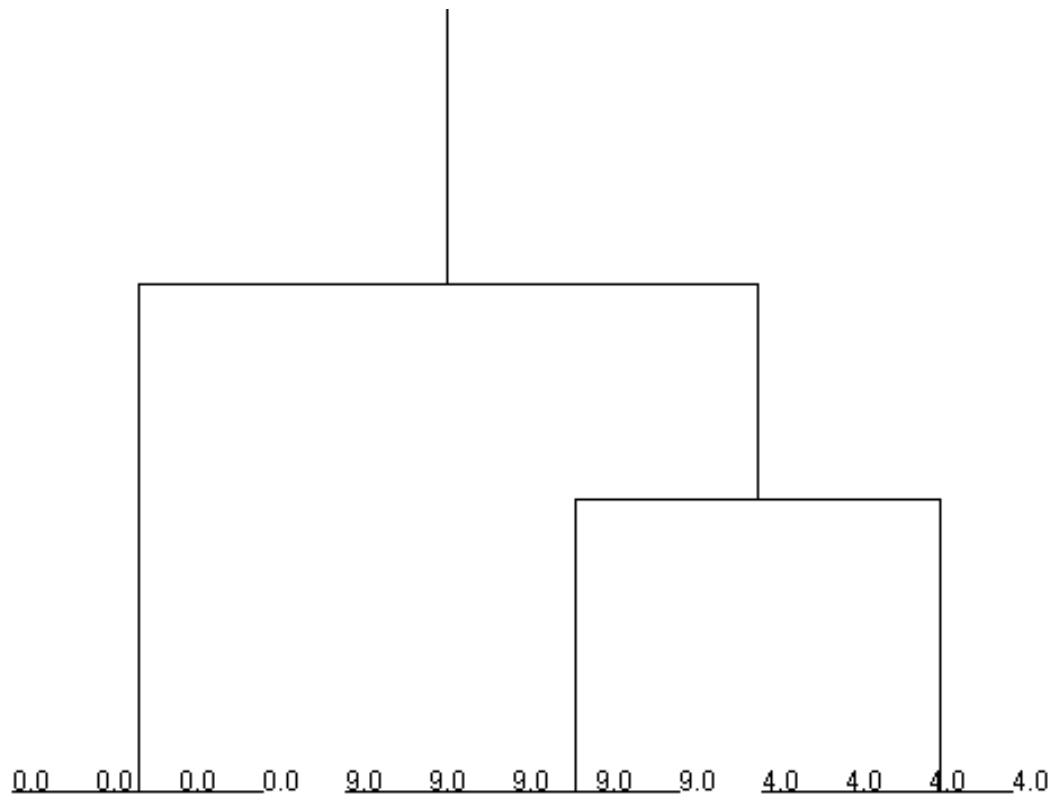


Figure 4.9: Dendrogram (Tree)

4.5.4.2 Kmeans Clustering

During kmeans clustering, the total number of clusters achieved was 70 and total number of samples (features) used is 284,771. This result was achieved after series of evaluation and applying different number of clusters, such as updating the number of clusters are done manually. The training dataset was in nominal form and K-means functions well when the data is in numerical. To accomplish this, the data was pre-process

and normalized by assigning or converting the nominal features into numerical automatically using Weka pre-process engine. Using this process, Weka reads the data as numerical and presents the result in a readable form as nominal (the original value). Refer to Appendix B for the clustering result.

4.5.4.3 IOC Clustering Result Discussion

Table 4.3 presents the clustering result of the two algorithms. Base on the table, Hierarchical clustering achieved less number of clusters compared to K-means. Hierarchical has the ability to automatically update its number of clustering, while K-means need to be done manually. The result from each clustering technique is used to create IOC logic and convert to IOCRule using OpenIOC framework.

Table 4.2: Comparison of Hierarchical and K-means Clustering Result

Parameters	K-Means	Hierarchical
No of Samples	284771	284771
No of Clusters	70	40

4.5.5 IOC Writer and IOCRule Creation

OpenIOC is used to write IOCRule structure from the clustered results. In order to develop an IOCRule, features of the malware files is selected from the data analysis stage and clustered after the feature extraction. Based on the cluster result, IOCRule is created, which replace or improve the traditional signature detection technique. The OpenIOC framework is used to structure the IOC logic and then manually restructures the format to Snort rule format. Figure 4.10 shows the IOCRule structure for Network Intrusion Detection system. A total number of 470 IOCRule were created from the kmeans

clustering analysis result and Hierarchical clustering analysis result. Figure 4.10 shows IOCRule Sample. Refer to Appendix B for the result of IOC logic using OpenIOC framework and IOCRule for NIDS.

```
#-----
# INDICATOR-COMPROMISE RULES
#-----

# alert tcp $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS (msg:"INDICATOR-COMPROMISE
.com- potentially malicious hostname"; flow:to_server,established; content:"Host:";
http_header; content:".com-"; within:70; fast_pattern; http_header; content:!"www";
within:3; distance:-8; http_header; pcre:"/Host: [^\x0d\x0a]*?\.com-/Hi";
metadata:policy security-ips drop, service http; classtype:bad-unknown; sid:32488;
rev:6;)

# alert tcp $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS (msg:"INDICATOR-COMPROMISE
c99shell.php command request - sql"; flow:to_server,established; content:"act=sql";
fast_pattern:only; http_uri; metadata:service http; reference:url,vil.nai.com/vil/
content/v_136948.htm; classtype:policy-violation; sid:16622; rev:5;)
```

Figure 4.10: IOCRule Sample

4.6 Experimental Study to Evaluate IOCRules

This section presents the experimental study of this thesis. It deals with the evaluation steps, result analysis and the result presentation. Figure 4.11 presents the evaluation pseudocode for this thesis.

Evaluation Pseudocode

```
Start
  initiate Snort in the running mode
  Deactivate all Signature Rules in the Snort Database
  Import Hierarchical Base IOCRule into a directory in the Snort Database
  Import K-means base IOCRule into a directory in the Snort Database
  Case 1:
    Select Hierarchical base IOCRule for evaluation
    if Hierarchical IOCRule is selected
    then; Deactivate K-means IOCRule
    else
      Select K-means base IOCRule for evaluation
    then; Deactivate Hierarchical base IOCRule
    Print Result
end
```

Figure 4.11: Evaluation Pseudocode

4.6.1 Experiment 1 : IOCRule Using Different Clustering Algorithm

Evaluation was conducted using equal number of IOCRule created from the clustered result of Hierarchical clustering and K-means clustering. A total number of 470 *IOCRule* were created from each clustering result, and implemented into IDS for the research evaluation, using CTU malware dataset.

4.6.1.1 Experimental Study Using IOCRule achieved from Hierarchical Clustering

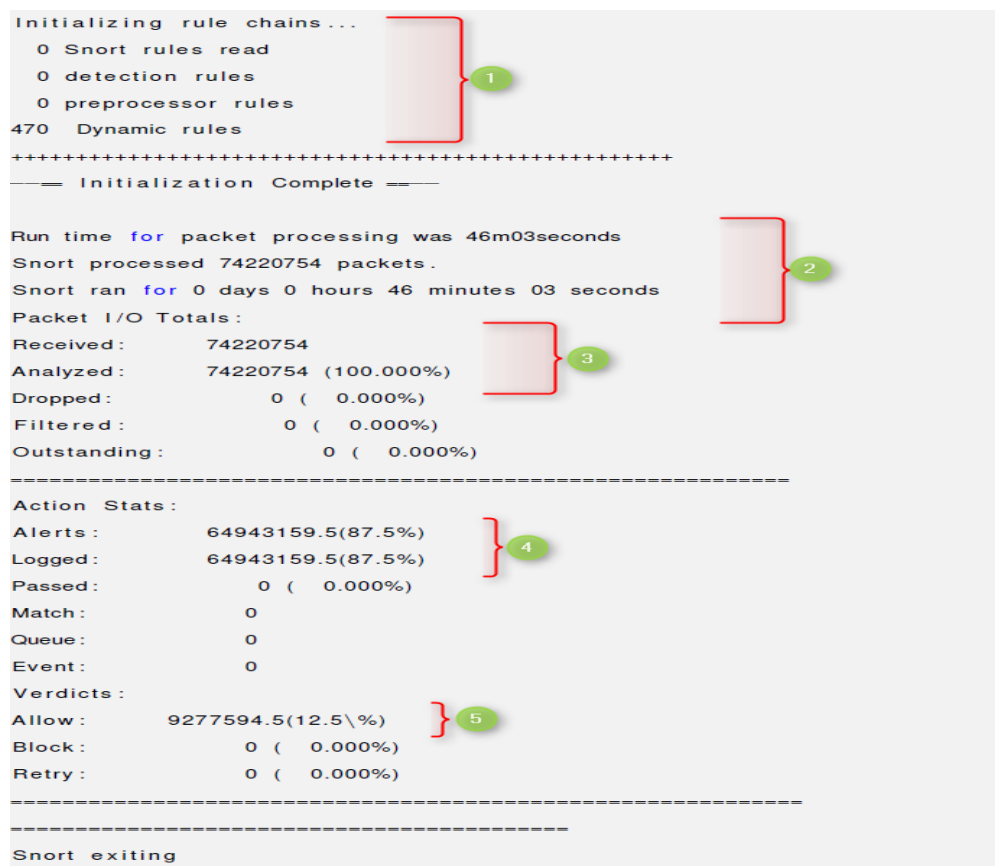


Figure 4.12: Experimental result of CTU Dataset on IOCRule (Hierarchical Clustering)

The snort first initialized the rules in the database and used them to detect intrusion from incoming traffic.

This indicated the run time for snort to process the DARPA datasets, and it showed total processed packets to be at 74,220,754, and it took 46 minutes and 03 seconds, which is referred to as the Detection Time.

This shows the total packets received by the detection engine. We can notice, it received the total packets processed in label 2. After it received the packets, it proceeded to 78 analyze them.

Presented the percentage of the alert generated and logged from the analyzed packets, which is at 87.5% from 100% of the packets, referred to as Detection Rate.

This label presented the number of intrusion packets allowed to pass the detection engine without the rules generating an alert, which was referred to as False Negative Rate.

4.6.1.2 Experimental Study Using IOCRule achieved from K-means Clustering

Figure 4.13 shows the result of experimental study using IOCRule achieved from K-means clustering.

```

Initializing rule chains...
470 Snort rules read
    0 detection rules
    0 decoder rules
    470 Dynamic rules
+++++
----- Initialization Complete -----
Run time for packet processing was 46m05seconds
Snort processed 74220754 packets.
Snort ran for 0 days 0 hours 46 minutes 05 seconds
Packet I/O Totals:
  Received:      74220754
  Analyzed:      74220754 (100.000%)
  Dropped:       0 ( 0.000%)
  Filtered:      0 ( 0.000%)
  Outstanding:   0 ( 0.000%)
  Injected:      0
=====
Action Stats:
  Alerts:        56045575.76 ( 75.512%)
  Logged:        56045575.76 ( 75.512%)
  Passed:        0 ( 0.000%)
Verdicts:
  Allow:         18175178.5 ( 24.488%)
  Block:         0 ( 0.000%)
  Retry:         0 ( 0.000%)
=====
Snort exiting

```

Figure 4.13: Experimental result of CTU Dataset on IOCRule

Step 1: The snort first initialized the rules in the database and used them to detect intrusion from incoming traffic.

Step 2: This showed the run time for snort to process the DARPA dataset, and it showed total processed packets to be 74,220,754 and it took 46 minutes and 05 seconds, which is referred to as the Detection Time.

Step 3: This showed the total packets received by the detection engine. We can notice, it received the total packets processed in label 2. After it received the packets, it proceeded to analyze them.

Step 4: Presented the percentage of the alert generated and logged from the analyzed packets, which is at 75% from 100% of the packets, refer to as Detection Rate.

Step 5: This label presents the number of intrusion packets allowed to pass the detection engine without the rules generating an alert, which is refer to as False Negative Rate.

4.6.1.3 K-means IOCRule Result Discussion

Based on the evaluation result in Table 4.4, IOCRule created from Hierarchical clustering achieved a better result in the evaluation study over the IOCRule from K-means clustering. This is argued based on the Detection Rate and False Negative Rate, as Hierarchical clustering has a higher detection of 87.5% and lower false negative of 12.5% as compared to 75.512% Detection rate of K-means and 24.488% of false negative. As explained in Chapter 3, Hierarchical clustering automatically assigns number of clusters until it achieves better number of clusters, while K-means clusters are assigned manually.

Table 4.3: IOCRule with different Clustering Algorithm

Parameters	Hierarchical	K-Means
No of IOCRule	470	
No of packets	74,220,754	
Detection Rate	64,943,159.5(87.5%)	56,045,575.76(75.512%)
Detection time	46m03s	46m05s
False Negative	9,277,594.5(12.5%)	18,175,178.5(24.488%)

4.6.2 Experiment 2: Vary Number of IOCRule

Table 4.4 presents the experimental result of this thesis using different approaches, focusing on the number of IOCRule for the dataset evaluation. The focus is to know the impact the number of IOCRule has over the detection effectiveness. The total IOCRule generated is 470, and evaluation is done using the 100%, 75%, 50% and 25% of 470 IOCRule. At the end of the evaluation a graphical representation to show the impact in different categories such as: Detection Rate, Detection Time and False Negative Rate at the discussion section.

Table 4.4: Vary number of IOC

Hierarchical /K-means						
No of IOCRule	Detection Rate		Detection Time		False Negative Rate	
	Hierarchical	K-means	Hierarchical	K-means	Hierarchical	K-means
470	87.5%	75.512%	46m03s	46m05s	12.5%	24.48%
353	81.02%	62.07%	46m02s	46m05s	18.88%	37.93%
235	68.47%	41.23%	38m42s	36m05s	31.53%	58.77%
118	58.76%	40.05%	38m01s	35m20s	41.24%	59.95%

4.6.2.1 Detection Rate

This section presents the graphical method, showing the Detection Rate of the experimental study (Vary Number of IOCRule). From the Figure 4.16, it can be concluded that as the number of IOCRule increases, the rate of detection also improved. For example, when IOCRule is equal to 470, the detection rate is 87.5% as compared to IOCRule with the value of 353 at 81% detection rate.

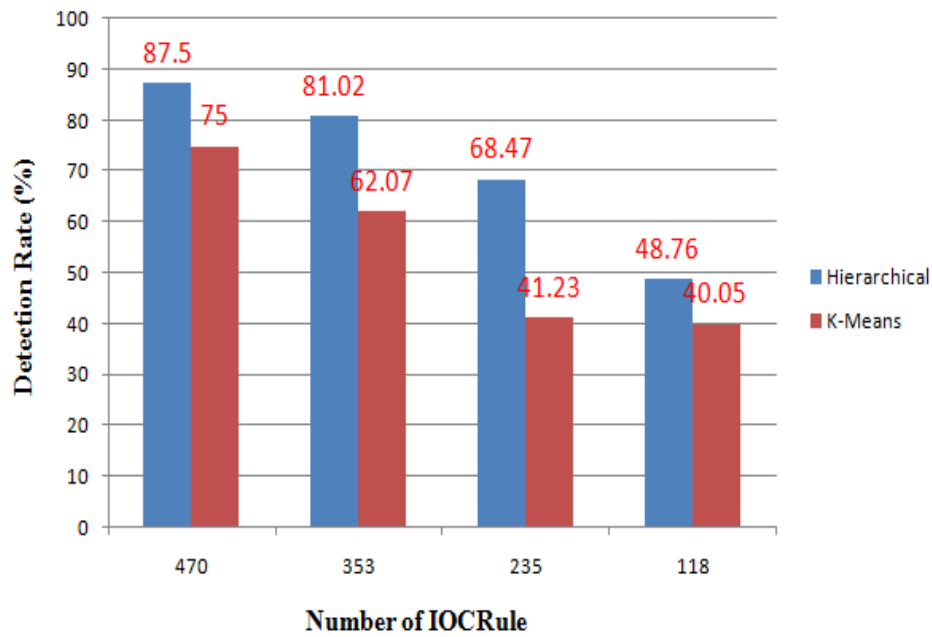


Figure 4.14: Detection Rate

4.6.2.2 Detection Time

This section presents the graphical method, showing the Detection Time of the experimental study (Vary Number of IOCRule). However, we note that as the number of IOCRule increased as shown in Figure 4.16, the time taken for detection also increased as shown in figure 4.17. For example detection rate of 470 IOCRule 87.5% which yielded a detection time of 46.02 minutes, compared to detection rate of 235 IOCRule 68.42% which yielded a detection time of 38.42 minute. Based on Figure 4.17, there is minimal differences in detection time between Hierarchical and K-means algorithm. Therefore, the choice should be to use Hierarchical since it shows higher detection rate compared to K-means algorithm.

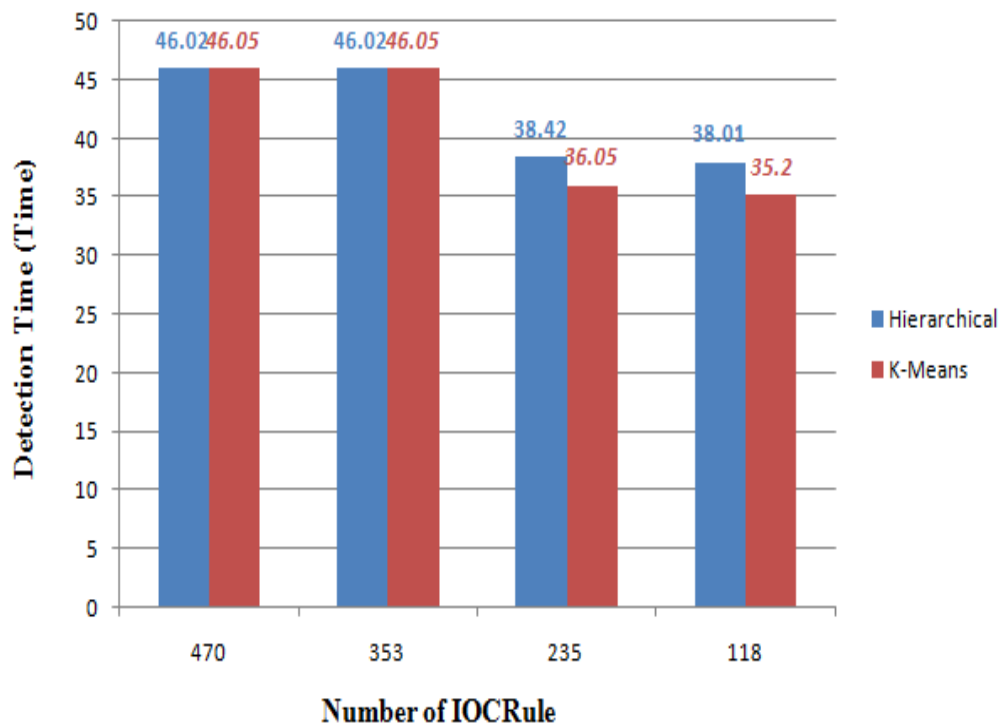


Figure 4.15: Detection Time

4.6.2.3 False Negative Rate

This section presents the graphical method, showing the false alarm rate of the experimental study (Vary Number of IOCRule). From Figure 4.18 the graph showed that higher number of IOCRule yields lower false negative rate. The conclusion can be said that, when the number of IOCRule is high, the low false negative rate result is achieved.

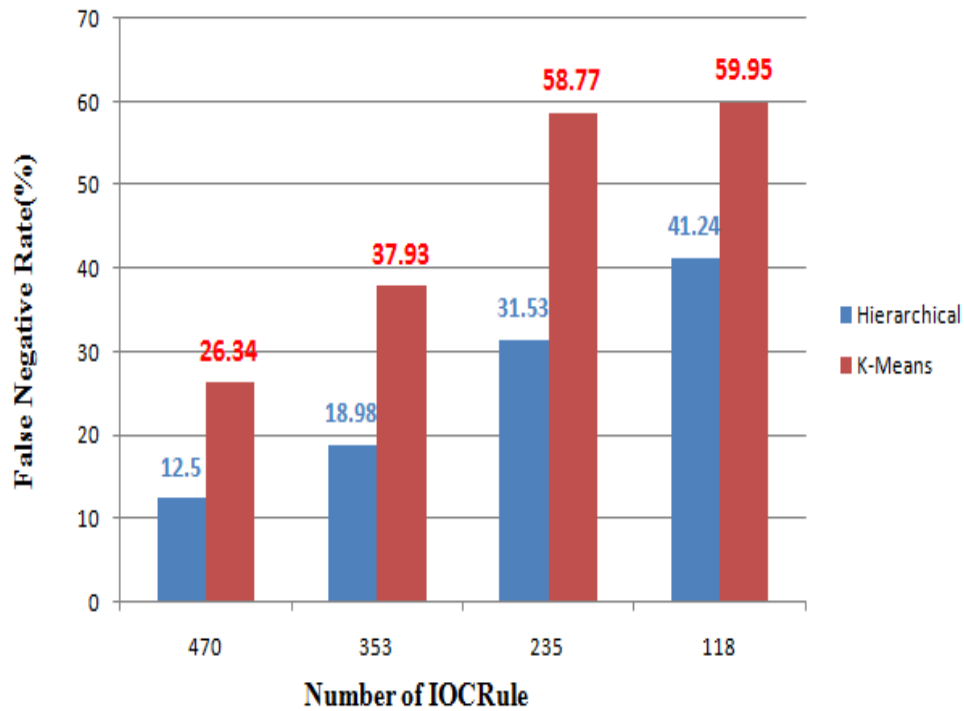


Figure 4.16: False Negative Rate

4.6.2.4 Experiment 2 Result Discussion

This section shows the evaluation result of varied number of IOCRule to determine the impact, size or number of IOCRule have in the performance detection of malicious activities. Section Experiment 2: Vary Number of IOCRule showed the evaluation using 470 IOCRule, 353 IOCRule, 235 IOCRule and 118 IOCRule for both Hierarchical and K-Means. The result achieved showed that Hierarchical has 87.4% detection rate, with 46 minutes 02 seconds detection time, 12.5% false negative rate, and K-means has 75.% detection rate, 46 minutes 05 seconds detection time and 26.34% false negative rate, when using 470 IOCRule.

This experiment was conducted three times in other to arrive at a conclusion. Based on experimental study result of different number of IOCRule, Figure 4.16 shows detection

of the clustering in different size of IOCRule, Figure 4.17 represent the Detection Time and Figure 4.18 the False Negative Rate.

4.6.3 Experiment 3: IOCRule with and without Clustering Algorithm

This sections presents result of experimental 3 study, which is the evaluation of IOCRule with and without clustering techniques. The aim of the experiment is to know the effectiveness of IOCRule without clustering techniques and the impact of clustering techniques in creating IOCRule.

Table 4.5: IOCRule with and without Clustering Algorithm

Parameters	IOCRule (Hierarchical)	IOCRule (Without Clustering)
No of IOCRule	470	
No of packets	74,220,754	
Detection Rate	64,943,159.5(87.5%)	1,147,153.303 (64.70%)
Detection time	46m03s	38m40s
False Negative	9,277,594.5(12.5%)	26,199,926.16(35.3%)

4.6.3.1 Experiment 3 Result Discussion

Table 4.7 presents the summary result of experiment 3. Table 4.7 presents the result of an evaluation study using an IOCRule created without the clustering algorithm process. The Detection Rate was at 64.70%, Detection Time at *38 Minutes 40 seconds* and a False Negative of 35.3%. Also it shows the result of the evaluation study using IOCRule with clustering algorithm process. It recorded a Detection Rate of 87.50%, Detection Time of *46 minutes 3 seconds* and a False Negative of 12.50%. A better result is achieved with an

IOCRule that went through clustering algorithm process, compared to IOCRule without clustering process.

Based on the result few things can be noted, clustering algorithm analysis and group the data into similar groups. This technique makes it easy to develop effective IOCRules, without having to duplicate the same rules unlike the IOCRule without clustering process. Though structure of IOCRules can differ but the features for the rules may belong to the same group family, making the IDS having rules that are inefficient due to multiples of same behavior Rules. The result presented in the table format is achieved from the evaluation done using IOCRule with clustering performed better compared to the evaluation without the clustering.

4.7 Comparative results discussion between the existing approach and the experimental study

The existing approach study which was carried out using DARPA and CTU datasets showed the effectiveness of Signature-Based IDS and the drawback in the aspect of False Negative on the detection of a malicious attack. The existing approach study showed signature-Based IDS had a base performance on malware dataset. Refer to Table 4.1, Section 4.4.3 for the evaluation result.

The first experimental study of this thesis is to determine, which clustering algorithm performs better in the clustering of IOC for IOCRule generation. The experimental result showed that hierarchical clustering algorithm performed better than K-means in this research. Refer to Table 4.4 Section 4.6.1.3 for the experimental result.

The third experimental study is to determine the impact of vary in IOCRule Size, which refers to the performance of different IOCRule size in malware detection. The

conclusion of this experiment can be said that the size of the IOCRule has an impact on the effectiveness of the IDS detection and the more the IOCRule the better the detection performance. Refer Table 4.5 and Table 4.6 Section 4.6.2.

The last evaluation determines the effectiveness of IOCRule with and without clustering algorithm process. Table 4.7 presents the result of IOCRule without clustering process and Table 4.7 presents with the clustering process. A conclusion is drawn from the two tables, which shows that the IOCRule with the clustering algorithm process performs better than the one without clustering, the base for the detection rate and false-negative rate.

4.7.1 Discussion on each Parameter

The equal number of malware dataset packets is used in the evaluation of the existing approach and proposed framework IOCRule using Hierarchical and K-means clustering techniques. Table 4.8 present the summary of the existing approach and the experiment result.

Table 4.6: Experimental Result

Parameters	Existing Approach	Hierarchical	K-Means
No of packets	74,220,759		
Detection Rate	890,811(1.184%)	64,943,159.5(87.5%)	55,665,565.5(75%)
detection time	1h43m40s	46m3s	46m05s
False Negative	73,329,882(98.816%)	9,277,594.5(12.5%)	19,555,189(26.34%)

4.7.1.1 Detection Rate

Figure 4.19 shows the comparative chart of the existing approach and proposed framework result. The detection rate result is presented in percentage format. The detection rate in this study shows the percentage of malware each method was able to detect from the dataset. Based on Figure 4.19 we can conclude and identify that the result of the evaluation carried out by the IOCRule created from the clustering algorithm achieved a better detection rate, compared to the existing approach, which is the tradition IDS. The existing approach evaluation shows a detection rate of 1.184%, while the proposed solution using Hierarchical clustering presents a detection rate of 87.5% and K-means presents a detection rate of 75%.

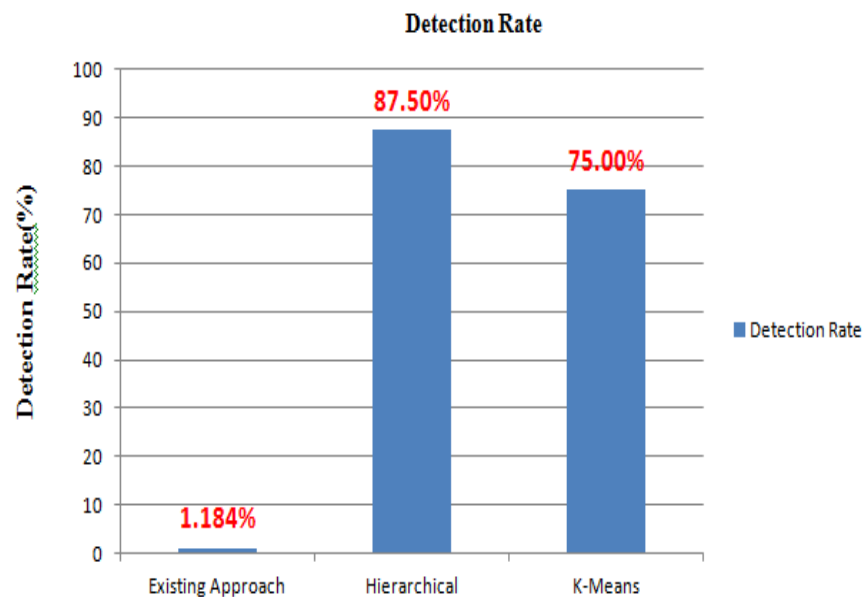


Figure 4.17: Detection Rate

4.7.1.2 Detection Time

The detection time, refers to the duration it took each evaluation method to detect malicious activities during the evaluation using CTU malware dataset. As presented in

Figure 4.20, the existing approach evaluation show the detection time of *1 hour 43 minutes 40 seconds*, while the proposed solution shows detection time of *46 minutes 30seconds* for Hierarchical clustering and detection time of *46 minutes 05 seconds* for K-means clustering. Traditional IDS (Signature-based) uses pattern method to match intrusion with signatures in the database, and it has to read every signature one after another to trigger an intrusion alert. At the other hand the proposed solution is a behavior approach to detection techniques. IOC are malware behaviors features that shows or are part of the life cycle of the malware.

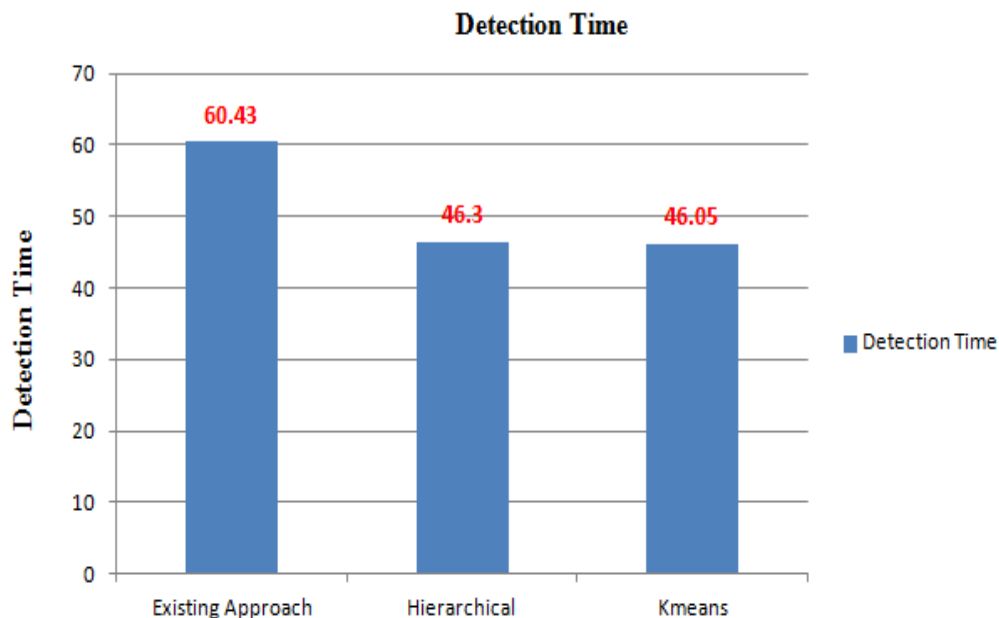


Figure 4.18: Detection Time

4.7.1.3 False Negative

Figure 4.21 shows the false negative rate of evaluation results of the existing approach and the proposed solution. False negative signifies a situation where an intrusion was undetected but flags as legitimate traffic. To have a low false negative, a good detection rate is needed and good detection rate is the outcome of a good method or

analysis of intrusion detection. Based on Figure 4.21 it can be identified that Hierarchical clustering achieved low false negative of 12.5% and K-means 26.34%, over the existing approach evaluation which achieved false negative of 98.816%. A conclusion can be drawn in this section that Hierarchical clustering performs better than K-Means, and the proposed solution performs better than the existing approach.

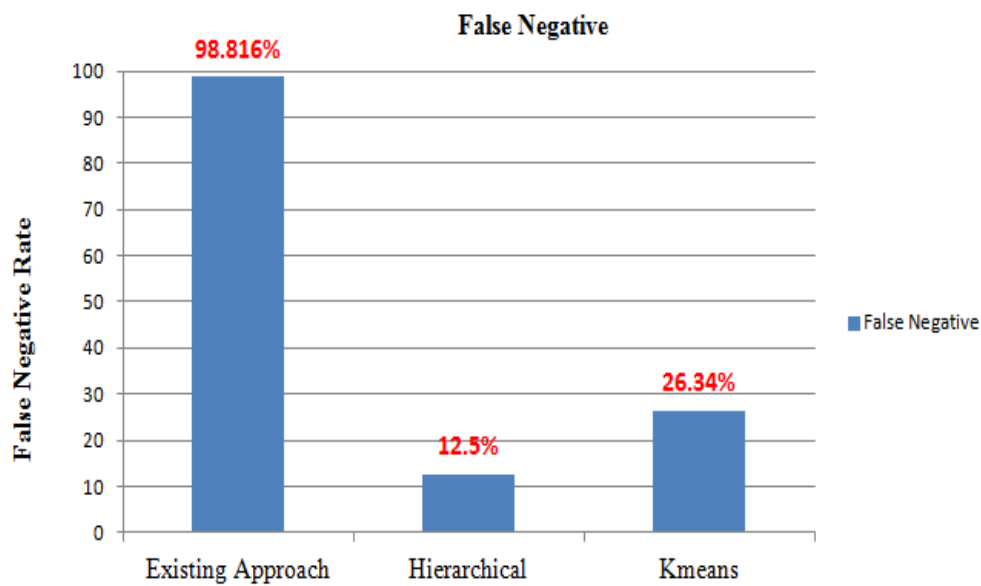


Figure 4.19: Number of False Negative

4.8 Result Conclusion

Based on the evaluations study, experimental study, and discussion sessions, this research has clearly shown the outcome of the thesis.

The evaluation study carried out on the Signature-Based IDS using an Old Dataset (DARPA Dataset), and a new malware dataset (CTU dataset) showed a remarkable result, which was identified that the Signature-Based IDS had a high detection rate on the old dataset and very low detection rate on the new dataset. A conclusion was made that, signature-based IDS did not have the capacity to detect malware attacks, due to the

behavior nature of malware and signature-based techniques function well when the signatures of the incoming attack is pre-generated and stored in the detection database. The first experiment was to identify the impact of clustering techniques in the generation of IOCRule in malware detection. Based on the result achieved from the study, a conclusion was made, stating that hierarchical clustering produced better IOCRule compares to K-means clustering.

The second experiment was to identify the impact number of IOCRule has in malware detection. The original number of IOCRule generated was 470, to conclude this study of varying the number of IOCRule, the 470 IOCRule considered as 100%, was reduced to 70%, 50% and 25%. Each of this percentage in the number of forms was used to conduct the study of variations in the number of IOCRule. A conclusion was made that the higher the detection Rule, the higher the detection rate, also the number of IOCRule has an impact on the detection time and false negative rate.

The third experiment was conducted to know the impact of on the IOCRule with and without clustering algorithm. The experiment was divided into two, and the first path is an experimental study to evaluate IOCRule with clustering techniques and the second path is to evaluate IOCRule without clustering techniques. Based on the results and the discussion sections, the conclusion showed that IOCRule with the clustering algorithms performed better in detection over IOCRule without the clustering algorithms. This performance can be said to occur due to the impact of the dynamic approach clustering algorithm applied by grouping the features in order not to have a repetition of features in more than one group, making the generated IOCRule effective and efficient compared to the IOCRule without clustering.

4.9 Summary

This chapter presented the framework design and the experimental study of the research. A conclusion was made by proving the effectiveness of the proposed framework over the existing method used in IDS. Evaluation study was done and conclusion arrived based on detection rate and false negative rate of the detection methods.

CHAPTER 5

CONCLUSION AND FUTURE WORK

This study has presented the overall issue of the Signature-Based IDS in malware detection environment and proposed the concept of Indicators of Compromise (IOC) to overcome the issue of signature detection. The main focus of this research is to address the issue of signature-based IDS malware detection using proposed behavior approach method of analysis and detection. There were three experimental study conducted in thesis to make conclusion of the result as explained in Chapter 4. The first experiment show that hierarchical clustering produced better IOCRule compares to K-means clustering. The detection rate achieved by the IOCRule from hierarchical clustering is 87.5% compare to K-means which is 75.51% detection and 24.48% false negative rate. The second experiment identified the impact number of IOCRule has on malware detection. A conclusion was made that the higher the detection Rule, the higher the detection rate, also the number of IOCRule has an impact on the detection time and false negative rate. The third experiment identified the impact of IOCRule with and without clustering techniques. Base on the result and the discussion sections, the conclusion drawn showed IOCRule with the clustering algorithms performs better in detection over IOCRule without the clustering algorithms. This performance can be said to occur due to the impact of the dynamic approach clustering algorithm applied by grouping the features in order not to have a repetition of features in more than one group, making the generated IOCRule effective and efficient compared to the IOCRule without clustering.

This thesis shows the proposed approach using IOCRule achieved a detection rate of 87.50%, false negative of 12.50% when evaluated using CTU 2016/2017 Malware dataset.

As the evaluation of CTU 2016/2017 Malware dataset achieved a detection rate of 1.184% and a false negative rate of 98.816%. This shows that the proposed approach achieved a much higher detection rate and lower false negative rate compared to the signature-based detection. The remainder of this chapter is organized as follows: Section 5.1 highlights the contributions made by this thesis. Section 5.2 highlights the challenges face during the research. Section 5.3 provides discussion on potential future works, which can be derived from the research works conducted in this thesis.

5.1 Summary of Contributions

The thesis has made several contributions as follows:

High Accuracy Network Intrusion Detection Method: Based on the high accuracy detection issue of Signature-Based IDS on malware attack, clustering algorithm was applied on malware behavior features to develop a dynamic IOCRule detection mechanism for malware attack. Hierarchical and K-means clustering were used to achieve high accuracy detection, as the clustering of the malware IOC data was an important aspect of grouping malware features into similar behavioral group to avoid duplication of IOCRule creation, which may result to delay in detection and false alarm.

A Dynamic Approach of Unknown Attack Detection: Signature-based IDS has issue on unknown attack detection, as the signature of such attack must exist in the signature database in order for an intrusion detection to occur. Based on the result gathered in this thesis on existing approach and the experimental study, there was a significant contribution made as the issue of unknown attack detection was overcome using a dynamic approach, which comprise of a dynamic development of IOCRule.

Extraction of IOC and Development of IOCRule for Malware Detection: Due to the dynamic nature of malware activities, a dynamic approach is needed for detection purpose. IOCRule was developed for malware detection, as IOCs are behavior features of malicious activities. This thesis extracted six (6) IOC for the development of IOCRule used for intrusion detection. Based on the research existing approach and experimental result, it showed that, the IOCRule achieved a significant result in malware detection and that justified the stated problem statement in Chapter One (1), over signature IDS.

A Dynamic Framework for the Detection Technique: The overall process of the research was achieved through a design framework, which includes the process of data collection, data analysis, feature (IOC) extraction, IOC clustering, IOCRule generation and the transfer of the IOCRule into IDS for detection.

5.2 Challenges

This section presents the challenges faced during this research.

Malware Data Collection: Data collection was one of the major challenges faced during the execution of this thesis. This was a challenge due to the university network access privilege, as many ports and services were blocked. This was a big challenge as the implemented honey-pot system was unable to collect any useful data for the benefit of the research. This challenge was overcome by setting up a private LAN which bypassed the university Network, and allowed the honey-pot to function as expected and was able to collect useful malware data for training purpose.

Malware Data Analysis: Due to the high memory usage during the analysis of the collected malware data from the honey-pot, a share of resources was needed to reduce the

high memory usage, as this was a challenge. To overcome it, the collected data was stored in a Synology and the analyzing system automatically fetches the data from the Synology.

5.3 Future Work

The research work presented in this thesis provides a basis for several potential related future works as follows:

Automatic Analysis Clustering of Malware Data: An automatic analysis where data has been fetched from the data collection system, extracting features automatically and clustering by a prediction method. This reduces the interaction of human for time efficiency, as time is an important factor in malware detection and analysis.

Machine Learning Prediction of Indicators of Attack (IOA): Indicators of Attack is a way to identify the attack before it occurs. This method of detection is needed to defend information or network against unknown or malicious code attack. On the other hand, machine learning helps to analyze and predict a situation. An implementation of a framework that should comprise of Machine learning to analyze and identify IOA and develop a prevention rule on a fly against emerging unknown attacks.

Implementation of IDS with behavior analysis and IOC/IOA extraction: It is of importance to develop IDS that can perform behavior analysis and IOC/IOA extraction. Every IOC/IOA extracted can be used for IDS rules or to analyze other attack systems to identify the cause or malware family of such intrusion.

REFERENCES

- [1] Bia, L., Liang, J., Dang, C., & Cao, F. (2012). A cluster centers initialization method for clustering categorical data. *Expert Systems with Applications*, 39(9), 8022–8029.
- [2] Zhao, D., Traore, I., Sayed, B., Lu, W., Saad, S., Ghorbani, A., & Garant, D. (2013, November 27). *Botnet Detection Based On Traffic Behavior Analysis And Flow Intervals*. Computer and Security, 2-16. doi: 10.1016/j.cose.2013.04.007.
- [3] Sangwan, O. P., & Vinod K. (2012). Signature Based Intrusion Detection System Using Snort. *International Journal of Computer Applications and Information Technology*, 1(3), 35–41
- [4] Sharma, N., Bajpai, A., & Litoriya, R. (2012). Comparison the various clustering algorithms of weka tools. *International Journal of Emerging Technology and Advanced Engineering*, 2(5), 73–80.
- [5] Anitha, E. S., Akilandeswar, J. & Sathiyabhama, B. (2011). A Survey on Partition Clustering Algorithms. *International Journal of Enterprise and Computing and Business Systems*, 1(1), 1–14.
- [6] Bailey, M., Oberheide, J., Andersen, J., Mao, Z.M., Jahanian, F., & Nazario, J. (2007). *Automated Classification and Analysis of Internet Malware*. Kruegel, C., Lippmann, R., Clark, A. (eds) Recent Advances in Intrusion Detection. RAID 2007. Lecture Notes in Computer Science, 4637. Springer, Berlin, Heidelberg. doi.org/10.1007/978-3-540-74320-0_10
- [7] Bayer, U., Kruege, I. C. & Kirda, E. (2006, May 11). *TTAnalyze: A Tool for Analyzing Malware*. In 15th European Institute for Computer Antivirus Research Annual Conference, Vienna, Austria. doi=10.1.1.60.7584
- [8] Snapp, S. R, Brentano, J., Nari, S. G. V Dias, & Goan, T .L (1991). *DIDS (Distributed Intrusion Detection System), Motivation, Architecture, and An Early Prototype*. In Proceedings of the 14th National Computer Security Conference.

- [9] Kao, C., Chang, Y., Nen-Fu, H., Rong-Tai, L., & Che-Wei, L. (2015, Sept. 23-25). *Automatic NIDS Rule Generating System for Detecting HTTP-like Malware Communication*. International Conference on Intelligent Information Hiding and Multimedia Signal Processing, Adelaide, SA, Australia. doi: 10.1109/IIH-MSP.2015.10.
- [10] Hesham, A. & Khalid, S. (2013). (WHASG) Automatic SNORT Signatures Generation by using Honeybot. *Journal of Computers*, 8(12), 3280–3286.
- [11] Sounak, P., & Bimal, K. M. (2014). Honeybot-based Signature Generation for Polymorphic Worms. *International Journal of Security and Its Applications*, 8(6), 101–114.
- [12] Bhatia, J. S., Sehgal, R. K. & Sanjeev, K. (2011). Botnet Command Detection using Virtual Honeybot. *International Journal of Network Security and Its Applications (IJNSA)* 3(5), 177–3189.
- [13] Verma, M., & Srivastava, M. (2014). A Comparative Study of Various Clustering Algorithms in Data Mining. *International Journal of Engineering Research and Applications*, 11(11), 422-428.
- [14] Brumley, D., Hartwig, C., Liang, Z., Newsome, J., Poosankam, P, Song, D., & Yin, H. (2007). Automatically identifying trigger-based behavior in malware: Botnet Analysis and Defense. In Wenke Lee (Ed.), *Botnet Analysis and Defense* (365-387). New York, NY: Science Press.
- [15] María, P. T., Tixeco, L. P., Pérez, G. S., & Medina, L. K. T. *Intrusion Detection Using Indicators of Compromise Based on Best Practices and Windows Event Logs*. The Eleventh International Conference on Internet Monitoring and Protection, 29–37. Retrieved from http://www.thinkmind.org/index.php?view=articlearticleid=icimp_2016_2_20_30032.
- [16] Padmini, R., & Nitin, J. (2013). Honeybot Technique Used For Intrusion Detection System. *International Journal of Science Engineering and Technology Research*. 2(12), 2204–2212.

- [17] Zalewski, M. (2001). *Strange Attractors and TCP/IP Sequence Number Analysis*. Retrieved from [http:// www.bindview.com](http://www.bindview.com).
- [18] Chaudhari, B. & Parikh, M. (2012). A Comparative Study of Clustering Algorithms Using Weka Tools. *International Journal of Application or Innovation in Engineering and Management*, 1(2), 154–158.
- [19] Gupta, J. & Mahajan, A. (2016). BPSO Optimized K-means Clustering Approach for Data Analysis. *International Journal of Computer Applications*, 133(15), 9–14.
- [20] Jeferson, W. S., & Luan, L. (2014). Network Traffic Monitoring and Analysis: PathanThe. In Al-Sakib Khan (Ed.), *The State of the Art in Intrusion Prevention and Detection* (23–46). Oxford, England: Berg.
- [21] Hong Kong Special Administrative Region, (2008, January 7). *Honeypot Security*. Retrieved from [http : / / www. infosec . gov. hk / english / technical / files / honeypots . pdf](http://www.infosec.gov.hk/english/technical/files/honeypots.pdf).
- [22] Levine, J., Bella, R., Owen, H., & Culver, B. (2012, June 18-20). *The Use of Honey Nets To Detect Exploited Systems Across Large Enterprise Networks*. Information Assurance Workshop, IEEE Systems, Man and Cybernetics Society, West Point, NY, USA. Doi: 10.1109/Smcsia.2003.1232406.
- [23] Kaur, R. & Singh, R. (2014). *Efficient Hybrid Technique for Detecting Zero-Day Polymorphic Worms*. Advance Computing Conference (IACC), 2014 IEEE International, Detroit, MI.
- [24] Dodge, R., Ragsdale, D. J., & Reynolds, C. (2003). *Organization and Training of a Cyber Security Team*. Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, Washington, DC, USA. doi:10.1109/ICSMC.2003.1245662.
- [25] Reddy, E. K. (2013). Neural Networks for Intrusion Detection and its Applications. *Proceedings of the World Congress on Engineering 2*. London, U.K.

- [26] Elavarasi, S. A., Akilandeswari, J., & Sathiyabhama, B. (2011). A Survey On Partition Clustering Algorithms. *International Journal of Enterprise Computing and Business Systems*, 1(1), 1–14.
- [27] Kim, H. A., & Karp, B. (2010). *Autograph: Toward Automated, Distributed Worm Signature Detection*. San Diego, CA: Proc. of 13 USENIX Security Symposium.
- [28] Feily, M., Shahrestani, A. & Ramadass, S. (2009). *A Survey of Botnet and Botnet Detection*. In SECURWARE '09: Proceedings of the 2009 Third International Conference on Emerging Security Information, Systems and Technologies, Athens, Glyfada, Greece. doi: 10.1109/SECURWARE.2009.48.
- [29] Fortinet (2017, August 20). *Fortinet Threat Landscape Report Q2 2017*. Retrieved from http://img03.en25.com/Web/Fortinet/%7Bb76df70f-3113-4eca-8b05-ef6887954e%7D_Fortinet-Threat-Report-Q2-2017.pdf.
- [30] Francois, G., Labiche, Y. & Massicotte, F. (2006, Dec. 11-15). *Automatic Evaluation Of Intrusion Detection Systems*. Computer Security Applications Conference, Miami Beach, FL, USA. doi: 10.1109/ACSAC.2006.15
- [31] Freiling, F. C., Holz, T. & Wicherski, G. (2007). *Botnet Tracking: Exploring a Root-Cause Methodology to Prevent Distributed Denial-of-Service Attacks*. In ESORICS, Milan, Italy. doi:10.1007/11555827_19
- [32] .Gadhiya, S., & Bhavsar, K. (2013). Techniques for Malware. *International Journal of Advanced Research in Computer Science and Software Engineering*, 3(4), 972– 975.
- [33] Gartner, T., Flach P., & Wrobel, S. (2003). On Graph Kernels: Hardness Results And Efficient Alternatives. *In Proceedings of the Sixteenth Annual Conference on Computational Learning Theory*, 16(9), 129-143.
- [34] Stinson, E., & Mitchell, C. (2007). Characterizing Bots Remote Control Behavior. *In Proceedings of the 4th GI International Conference on Detection of Intrusions and Malware and Vulnerability Assessment (DIMVA '07)*. Berlin, Heidelberg

- [35] Christian, K. & Jon, C. (2014). Honeycomb – Creating Intrusion Detection Signatures Using Honey pots. *ACM Sigcomm Computer Communications Review*, 34(1), 51– 56.
- [36] Hanieh, R., Muhammad, N. M. & Alireza, M. (2010). *A Framework for Automated Malcode Signatures Generation*. Proceedings of 2010 IEEE Student Conference on Research and Development (SCORED 2010), Putrajaya, Malaysia. doi: 10.1109.2010.5703974.
- [37] Bellard, F. (2005). *A Fast And Portable Dynamic Translator*. In Usenix Annual Technical Conference, CA, USA.
- [38] Holz, T., Steiner, M. Dahl, .F., & Biersack, .E. (2008). *Measurements and Mitigation of Peer-to-Peer-Based Botnets: A Case Study On Storm Worm*, Cambridge. USENIX: In: Proceedings of the 1st USENIX Workshop On Large-scale Exploits and Emergent Threats. MA, USA.
- [39] Daley, K., Larson, R., & Dawkins, J. (2002). *A Structural Framework for Modeling Multi-Stage Network Attacks*. Proceedings of the International Conference on Parallel Processing Workshops (ICPPW'02), Vancouver, BC, Canada, Canada. Doi:10.1109/ICPPW.2002.1039705.
- [40] Hosmer, C. (2008, May 19). *Polymorphic and metamorphic malware*. Retrieved from [http://www.madlab. it/papers/www2016.pdf](http://www.madlab.it/papers/www2016.pdf).
- [41] Hun-Ya, L. (2013, Sept. 17). *Using IOC (Indicators of Compromise) in Malware Forensics*. Retrieved from <http://www.mandiant.com/news/release/mandiant-releases-openioc-standard-for-sharing-threatintelligence/>.
- [42] Mahoney, M., & Chan, P. (2009). *PHAD: Packet Header Anomaly Detection for Identifying Hostile Network Traffic*. Florida, Melbourne: Florida Inst. of Technology.
- [44] Jaina, P., & Krunal, P. (2015). Effective Intrusion Detection System using Data Mining Technique. *Journal of Emerging Technologies and Innovative Research (JETIR)*, 2(6), 1867–1878.

- [45] Ciza, T., Vishwas, S. & Balakrishnan, N. (2014, May 26). *Usefulness of DARPA Dataset for Intrusion Detection System Evaluation*. Retrieved from [http://core.kmi.openac.uk/download /pdf/11632149.pdf](http://core.kmi.open.ac.uk/download/pdf/11632149.pdf)
- [46] Teresa, F .L, Jagannathan, R., Lee, R., Listgarten, R., Edwards, D. L., Neumann, P.G., Javitz, H.S, & Valdes, A. (1988). IDES: The enhanced prototype: *A Real-Time Intrusion-Detection Expert System*. Technical Report SRI-CSL-88-12, SRI International, 333 Ravenswood Avenue, Menlo Park, CA.
- [47] Jyothsna, V., Rama, V. V., & Munivara, K. P. (2011). A Review of Anomaly based Intrusion Detection Systems. *International Journal of Computer Applications*, 28(7), 26–35.
- [48] Bekerman, D., Shapira, B., Rokach, L., & Bar, A. (2015). *Unknown Malware Detection Using Network Traffic Classification*. IEEE Conference on Communications and Network Security, Florence, Italy. doi:10.1109/CNS.2015.7346821.
- [49] Deris, S., Ala I. S., Yazid, I.,& Kamarulnizam, A. B. (2012). Intrusion Prevention System: A Survey. *Journal of Theoretical and Applied Information Technology*, 40(1), 44–54.
- [50] Kelly, K. (Feb., 2017). *Indicators of Compromise TeslaCrypt Malware*. Retrieved from 2017 The SANS Institute.
- [51] Khan, S. S., & Ahmad, A. (2013). Cluster center initialization algorithm for K-modes clustering. *Expert Systems with Applications*, 40(18), 7444–7456.
- [52] Fanfara, .P, Dufala, M. & Radušovský, J (2013). Autonomous Hybrid Honeypot as the Future of Distributed Computer Systems Security. *Acta Polytechnica Hungarica*, 10(4), 25–42.
- [53] Mohssen, Z., Eisa, A., & Neco, V. (2014b). Zero-day Polymorphic Worms Detection Using Aho-Corasick Algorithm. *Advances in Information Science and Computer Engineering*, 8(2), 211–216.

- [54] De Ocampo, Frances, B. C., Del Castillo, Trisha, M. L., Gomez, M., & Miguel, A. N. (2013). Automated Signature Creator For A Signature Based In-trusion Detection System With Network Attack Detection Capabilities (Pancakes). *International Journal of Cyber-Security and Digital Forensics*, 2(013), 25– 35.
- [55] Reshma, R., & Chirag, S. (2011). *Zero-Day Attack Signatures Detection Using Honey-pot*. International Conference on Computer Communication and Networks, Wuhan, China. doi: 10.1109/PDCAT.2010.54.
- [56] Liao, X., Yuan, K., Wang, .Z., Xing, L., & Beyah, R. (2016). *Acing the IOC game: Toward automatic discovery and analysis of open-source cyber threat intelligence*. In ACM Conference on Computer and Communications Security, Vienna, Austria.
- [57] Lincoln, MIT (2014). *DARPA: Intrusion Detection System Evaluation data set*. Retrieved from <http://www.ll.mit.edu/mission/communications/cyber/CSTcorpora/ideval/index.html>.
- [58] Maggi, F., Matteucci, M., & Zanero, S. (2009). Reducing false positives in anomaly detectors through fuzzy alert aggregation. *Information Fusion*, 10(4), 300–311.
- [59] Jack, K. (2003). *Intrusion Detection with Snort*. United Ststes, America: Sams Publishing.
- [60] Mahoney, V. (2003). *A Machine Learning Approach To Detecting Attacks by Identifying Anomalies in Network Traffic*. Melbourne, MA.
- [61] Caballero, J., Yin, H., Liang, Z., & Song, D. (2007). *Polyglot: Automatic Extraction of Protocol Message Format using Dynamic Binary Analysis*. In Proceedings of ACM Conference on Computer and Communication Security, Alexandria, Virginia, USA. doi:10.1145/1315245.1315286.
- [62] Mesakar, S. S., & Chaudhari, M. S. (2016, May 17) *A Review of Clustering Algorithms*. Retrived from [http://www.kaspersky.com/about/news/virus/2010/First SMS Trojan detected for smartphones running Android](http://www.kaspersky.com/about/news/virus/2010/First_SMS_Trojan_detected_for_smartphones_running_Android).

- [63] Mohssen, M. Z., Anthony, H. C., Neco, V. M., & Eihab, B. (2011). An Automated Signature Generation Approach for Polymorphic Worms Using Principal Component Analysis. *International Journal for Information Security Research*, 1(2), 45–52.
- [64] Mohssen, M. Z., Eisa, A., & Neco, V. (2014a). An Automated Signature Generation Method for Zero-day Polymorphic Worms Based on C4.5 Algorithm. *The Ninth International Conference on Software Engineering Advances*, 7(4), 215–220.
- [65] Kothari, C. R. (2004). *Research Methodology: Methods and Techniques*. New Age International.
- [66] Neha, S., & Nirmal, G. (2015). K-modes Clustering Algorithm for Categorical Data. *International Journal of Computer Applications*, 127(17), 1–6.
- [67] Novak, J., & Northcutt, S. (2003). *Network Intrusion Detection*. Retrieved from: <https://books.google.com.my/books>.
- [68] Carbonera, J. L. & Abel, M. (2014). *Categorical Data Clustering: A Correlation-Based Approach for Unsupervised Attribute Weighting*. In Proceedings of ICTAI, Detroit, MI.
- [69] Patil, R., Deshmukh, S., & Rajeswari, K. (2015). Analysis of Simple K-Means with Multiple Dimensions using WEKA. *International Journal of Advanced Research in Computer Science and Software Engineering*, 110(1), 14–17.
- [70] Peng, N., Yun, C., Douglas, R., & Dingbang, X. (2004). Techniques and Tools for Analyzing Intrusion Alerts. *ACM Transactions on Information and System Security*, 7(2), 274–318.
- [71] Phillip, A. P., & Alfonso, V. (1998, June 10). San Diego, CA: Internet Society.
- [72] University, CTU. *Malware Capture Facility Project*. Retrieved from <https://mcfpweebly.com/the-ctu-13-dataset-a-labeled-dataset-with-botnet-normal-and-background-traffic.html>.
- [73] Project, Deutsche Telekom AG Honeypot (2015). *T-Pot: A Multi-Honeypot Platform*. Retrieved from <http://dtag-dev-sec.github.io/mediator/feature/2015/03/17/concept.html>.
- [74] Quttera (2017). *Malicious Domains Database*. Retrieved from <https://quttera.com/lists/malicious>.

- [75] Rafique, M.Z., & Caballero, J. (2013). FIRMA: Malware Clustering and Network Signature Generation with Mixed Network Behaviors. In: Stolfo S.J., Stavrou A., Wright C.V. (Eds.) Research in Attacks, Intrusions, and Defenses. RAID 2013. *Lecture Notes in Computer Science*, vol 8145. Springer, Berlin, Heidelberg.
- [76] Rajesh, K., & Tajinder, K. (2014). Machine Learning based Traffic Classification using Low Level Features and Statistical Analysis. *International Journal of Computer Applications (0975 – 8887)*, 108(12), 6–13.
- [77] Ranjan, R., & Sahoo, G. (2014). A New Clustering Approach For Anomaly Intrusion Detection. *International Journal of Data Mining and Knowledge Management Process*, 4(2), 29–38.
- [78] Dornseif, M., Holz, .T. & Klei, C. (2004). *NoSeBrEaK: Attacking Honeynets*. Proceedings of the 5th IEEE Information Assurance Workshop, NY, USA. doi:10.1109/IAW.2004.1437807.
- [79] Leyla, B., & Tudor, D. (2012). *Before We Knew It: An Empirical Study of Zero-Day Attacks in the Real World*. ACM Conference on Computer and Communication Security, Raleigh, North Carolina, USA. doi:10.1145/2382196.2382284.
- [80] Robert, G. (2014). *FAQ: Network Intrusion Detection Systems*. Retrieved from mailto:nids-faq1@RobertGraham.com.
- [81] Roy, D. K., & Sharma, L. K. (2010). Genetic k-Means Clustering Algorithm For Mixed Numeric And Categorical Data Sets. *International Journal of Artificial Intelligence And Applications*, 1(2), 23–28.
- [82] Anuar, N. B., Papadaki, M., Furnell, S., & Clarke N. (2010). An Investigation and Survey of Response Options for Intrusion Response Systems (IRSs). *Information Security for South Africa (ISSA)*, 7(2), 1–8.

- [83] Shabtai, A., Menahem, E., & Elovici, Y. (2011). F-Sign: Automatic, Function-Based Signature Generation for Malware. *IEEE Transactions On Systems and Cybernetics*, 41(03), 494–508.
- [84] Bacciu, D., Jarman, I. H., Etchells, T. A., & Lisboa, P. G. (2009). *Patient Stratification with competing risks by multivariate Fisher distance*. International Joint Conference on Neural Networks, Atlanta, GA, USA. doi: 10.1109/IJCNN.2009.5179077.
- [85] Shirey, R. (2000, April 20). *RFC 2828: Internet Security Glossary*. New York: The Internet Society. Retrieved from <https://tools.ietf.org/html/rfc792>
- [86] Shiri, F. I., Shanmugam, B., & Idris, N. B. (2011). A Parallel Technique for Improving the Performance of Signature-Based Network Intrusion Detection System. In: *Communication Software and Networks*, 3(19), 692 –696.
- [87] Bayer, U., Milani, C., P., Hlauschek, C., Krugel, C., & Kirda, E. (2013). Scalable, Behavior-Based Malware Clustering. *International Journal of Computer Applications*, 28(8), 26–35.
- [88] Sophos (2014). *2014 Sophos-security threat report*. Retrieved from <http://www.sophos.com/en-us/medialibrary/pdfs/other/sophos-security-threat-report-2014.pdf>.
- [89] Bhagat, P .M, Halgaonkar, P .S. & Wadhai, V .M. (2013). Review of Clustering Algorithm for Categorical Data. *International Journal of Engineering and Advanced Technology*, 3(2), 341–345.
- [90] Grizzard, J. B., Sharma, V., & David, D. (2007). *Peer-to-Peer Botnets: Overview and Case Study*. Cambridge, MA Berkeley. USENIX: Proceedings of The First Workshop on Hot Topics in Understanding Botnet, CA, USA.
- [91] Subaira, A. S., & Anitha, P. (2013). An Efficient Classification Mechanism For Net-work Intrusion Detection System Based on Data Mining Techniques: A Survey. *International Journal of Computer Science and Business Informatics*, 6(1), 1–12.
- [92] Syal, R., & Kumar, V. V (2012). Innovative Modified K-Mode Clustering Algorithm. *International Journal of Engineering Research and Applications (IJERA)*, 2(4), 390–398.

- [93] Syarif, Y. S., Yudi, P., & Imam, R. (2015). Implementation of Malware Analysis using Static and Dynamic Analysis Method. *International Journal of Computer Applications*, 117(6), 11–15.
- [94] Tejvir, K., & Sanmeet, K. (2013). *Comparative Analysis of Anomaly Based and Signature Based Intrusion Detection Systems Using PHAD and Snort*. Retrieved from www.cse.iitk.ac.in.
- [95] Jensi, R., & Jiji, G .W. (2015). Hybrid Data Clustering Approach Using K-Means and Flower Polltination Algorithm. *Advanced Computational Intelligence: An International Journal*, 2(2), 15–25.
- [96] Tillmann, W., Fuchs, C., Gerhards-Padilla, E., & Martini, P. (2009). *Nebula - Generating Syntactical Network Intrusion Signatures*. 4th International Conference on Malicious and Unwanted Software (MALWARE), Bonn, Germany.
- [97] Toussain, M. *Home Field Advantage - Using Indicators of Compromise to Hunt down the Advanced Persistent Threat*. Retrieved from <https://www.sans.org/reading-room/whitepapers/detection/home-field-advantage-indicators-compromise-hunt-down-advanced-persistent-threat-35462>.
- [98] Postel, J. (2016, April 26). *RFC792: Networking Group: Internet Control Message Protocol: DARPA Internet Program Protocol Specification*. Retrieved from <https://tools.ietf.org/html/rfc792>.
- [99] Broder, A. Z., Glassman, S. C., Manasse, M. S., & Zweig, G. (1997). Syntactic clustering of the web. *Computer Network ISDN System*, 29(8-13), 1157–1166.
- [100] Vipin, D., & Vijaya, P. (2010). Network Intrusion Detection System Based On Machine Learning Algorithms. *International Journal of Computer Science and Information Technology (IJCSIT)*, 2(6), 138–151.
- [101] Whitepaper, SecBI (April, 2017). *Using Cluster Analysis for Comprehensive Threat Detection*. Retrieved from <http://www.secbi.com>.

- [102] Younghee, P., Qinghua, Z., Douglas, R., & Vikram, M. (2010). AntiBot: Clustering Common Semantic Patterns for Bot Detection. *IEEE 34th Annual Computer Software and Applications Conference*, 262–272. doi:10.1109/COMPSAC.2010.33
- [103] Catakoglu, O., Balduzzi, M. & Balzarotti, D. (2016, June 17). *Automatic Extraction of Indicators of Compromise for Web Applications*. Retrieved from: <http://www.madlab.it/papers/www2016.pdf>.
- [104] Ankita, B., Sonali, B. & Amit, P. (2015). Botnet Identification System Using Clustering and Machine Learning C5.0. *International Journal of Computer Science and Mobile Computing*, 4(4), 313–316.

APPENDICES

Appendix A: Snort Implementation

A.1 Install pre requisites for compiling snort

1. **sudo apt-get update**

This command updates ubuntu operating system. It is done in case some features in the operating system is not up to date.

2. **sudo apt-get install flex bison build-essential checkinstall \libpcap-dev libnet1-dev libpcr3-dev libmysqlclient15-dev \libnetfilter-queue-dev iptables-dev**

After the update is done, the above command will be entered to install prerequisites for compiling snort.

A.1.1 Download DAQ, Snort, Libdnet source code

a. Download DAQ

Data Acquisition (DAQ) is a snort package that has some modules implemented in it for packet captureing and packet drop. Some of the DAQ modules include:

- Pcap: the default mode, used for sniffer and IDS modes.
- Afpacket: inline on Linux using two bridged interfaces
- Dump: allows testing of inline and normalization mechanisms

The following command is used in downloading DAQ:

wget <https://www.snort.org/downloads/snort/daq-2.0.4.tar.gz>

b. Download Snort

Snort is an open-source Intrusion Detection System (IDS) that runs on Linux, UNIX, BSD variants and Windows. The following command is used in downloading Snort: **wget https://www.snort.org/downloads/snort/snort-2.9.7.0.1.tar.gz**

c. Download Libdnet Source Code

Libdnet provides a simplified, portable interface to several low-level networking routines, including network address manipulation, kernel arp(4) cache and route(4) table lookup and manipulation, network firewalling (IP filter, ipfw, ipchains, pf, PktFilter etc), network interface lookup and manipulation, IP tunnelling (BSD/Linux tun, Universal TUN/TAP device) and Raw IP packet and Ethernet frame transmission. **wget http://libdnet.googlecode.com/files/libdnet-1.12.1.tgz** command is use in downloading libdnet.

A.1.2 Unzip, Make and install Libdnet, DAQ and Snort

In linux operating system, after a software has been download, the next procedure is to make and install it into the system. The subsection of this section will show the process of the installation.

. Libdnet

The following steps are use to unzip, make and install libdnet.

```
tar zxvf libdnet-1.12.tgz  
cd libdnet-1.12/  
./configure CFLAGS = -fPIC;make  
sudo checkinstall  
sudo dpkg -i libdnet_1.12-1_i386.deb
```

```
sudo ln -s /usr/local/lib/libdnet.1.0.1 /usr/lib/libdnet.1
```

When you type **sudo checkinstall** will encounter The following messages:-

Should I Create a default set of package docs?

Type y and Enter

Do you want me to list them? [n]: n

Should I exclude them from the package? (Saying yes is a good idea) [n]: y

b. DAQ

The following steps are use to unzip, make and install DAQ.

```
tar xvfz daq-20.2.tar.gz
```

```
cd daq-2.0.2
```

```
./configure CFLAGS = -fPIC; make
```

```
sudo checkinstall
```

```
sudo dpkg -i daq_2.0.4-1_i386.deb
```

When you type **sudo checkinstall** will encounter The following messages Should I Create a default set of package docs?

Type y and Enter

Do you want me to list them? [n]: n

Should I exclude them from the package? (Saying yes is a good idea) [n]: y

A.2 Snort

The following steps are use to unzip, make and install Snort.

```
tar xvfz snort-2.9.7.0.tar.gz  
cd snort-2.9.7.0  
./configure --enable-sourcefire; make  
sudo checkinstall  
sudo dpkg -i snort_2.9.7.0-1_i386.deb  
sudo ln -s /usr/local/bin/snort/usr/sbin/snort  
sudo ldconfig -V
```

When you type **sudo checkinstall** will encounter The following messages Should I Create a default set of package docs?

Type y and Enter

Do you want me to list them? [n]: n

Should I exclude them from the package? (Saying yes is a good idea) [n]: y

A.2.1 Start Snort

To start snort the following command is entered (**snort -V**). And the following output is received after entering the command. Refer to Figure A.1:

```

rene@ubuntu:~$ snort -V

  , _
 o" )~
  ' '

-*> Snort! <*-
Version 2.9.7.0 GRE (Build 149)
By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
Copyright (C) 2014 Cisco and/or its affiliates. All rights reserved.
Copyright (C) 1998-2013 Sourcefire, Inc., et al.
Using libpcap version 1.5.3
Using PCRE version: 8.31 2012-07-06
Using ZLIB version: 1.2.8

rene@ubuntu:~$ █

```

Figure A.1: Start Snort

A.2.2 Snort Settings

The following command is used in the setting of snort user login; creating a file for rules and making a directory where the snort rules will be stored.

```
sudo groupadd snort
```

```
sudo useradd snort -d /var/log/snort -s /sbin/nologin -c SNORT_IDS -g snort
```

```
sudo mkdir /var/log/snort
```

```
sudo chown snort:snort /var/log/snort
```

```
sudo mkdir /etc/snort
```

After the above command has been entered, we downloaded the original file for snort

signature based rule from the snort website **www.snort.org** and the following command is used in the unzip and installation.

```
sudo tar zxvf snortrules-snapshot-2962.tar.gz -C /etc/snort/
```

```
sudo touch /etc/snort/rules/white_list.rules /etc/snort/rules/black_list.rules
```

```
sudo mkdir /usr/local/lib/snort_dynamicrules
```

```
sudo chown -R snort:snort /etc/snort/*
```

```
sudo mv /etc/snort/etc/* /etc/snort/
```

After the above command is entered, we edit and configure the snort in other for it to identify the host machine and enable to write our own signature rule to detect packets.

```
# Setup the network addresses you are protecting
```

```
# ipvar HOME_NET any
```

```
ipvar HOME_NET 192.168.8.0/24
```

```
# Here you need to check your network configure by using ifconfig
```

```
# Set up the external network addresses. Leave as "any" in most situations
```

```
# ipvar EXTERNAL_NET any
```

```
ipvar EXTERNAL_NET !$HOME_NET
```

```
# Path to your rules files (this can be a relative path)
```

```
# Note for Windows users: You are advised to make this an absolute path,
```

```
# such as: c:\snort\rules
```

```
var RULE_PATH /etc/snort/rules
```

```
var SO_RULE_PATH /etc/snort/so_rules
```

```
var PREPROC_RULE_PATH /etc/snort/preproc_rules
```

```
# If you are using reputation preprocessor set these
```

```
var WHITE_LIST_PATH /etc/snort/rules
```

```
var BLACK_LIST_PATH /etc/snort/rules
```

The next step is to start snort with the following command (**sudo snort -T -i eth0 -u snort -g snort -c /etc/snort**) and the installation and configuration will be done. After the start the following output is received. Refer to Figure A.2:

```
Copyright (C) 1998-2013 Sourcefire, Inc., et al.
Using libpcap version 1.5.3
Using PCRE version: 8.31 2012-07-06
Using ZLIB version: 1.2.8

Rules Engine: SF_SNORT_DETECTION_ENGINE Version 2.4 <Build 1>
Preprocessor Object: SF_DNP3 Version 1.1 <Build 1>
Preprocessor Object: SF_POP Version 1.0 <Build 1>
Preprocessor Object: SF_MODBUS Version 1.1 <Build 1>
Preprocessor Object: SF_GTP Version 1.1 <Build 1>
Preprocessor Object: SF_DNS Version 1.1 <Build 4>
Preprocessor Object: SF_SIP Version 1.1 <Build 1>
Preprocessor Object: SF_SSH Version 1.1 <Build 3>
Preprocessor Object: SF_IMAP Version 1.0 <Build 1>
Preprocessor Object: SF_DCERPC2 Version 1.0 <Build 3>
Preprocessor Object: SF_FTPTELNET Version 1.2 <Build 13>
Preprocessor Object: SF_SSLPP Version 1.1 <Build 4>
Preprocessor Object: SF_SMTP Version 1.1 <Build 9>
Preprocessor Object: SF_REPUTATION Version 1.1 <Build 1>
Preprocessor Object: SF_SDF Version 1.1 <Build 1>

Snort successfully validated the configuration!
Snort exiting
```

Figure A.2: Start Snort Successfully

Snort is now ready to use.

Appendix B Framework Design

B.1 Data Analysis and IOC Extraction

```
"id": 1,
"server_ip": "161.47.7.14:80",
"url": "/installer.php?ab=true&trackid=tracking=4&campaign=4&sessionid=b60b1200-2d14-4923-b640-40905c1b6d11&version=1539",
"short_url": "/installer...rsion=1539",
"req_head": "GET /installer.php?ab=true&trackid=tracking=4&campaign=4&sessionid=b60b1200-2d14-4923-b640-40905c1b6d11&version=1539 HTTP/1.1",
"req_body": "1",
"res_base64": "MQ==",
"respeek": "1",
"magic_name": "Inconclusive. Probably text",
"magic_ext": "TEXT",
"res_head": "HTTP/1.1 200 OK\r\nServer: Apache/2.2.15 (CentOS)\r\nVary: Accept-Encoding\r\nCache-Control: no-cache, must-revalidate\r\nContent-Type: text/html",
"res_num": "200 OK",
"res_type": "text/html",
"host": "www.reimageplus.com",
"referrer": "",
"filename": "installer.php",
"method": "GET",
"epochtime": "11/21/77 02:13:13",
"res_len": "1.0 B",
"md5": "c4ca4238a0b923820dc509a6f75649b",
"sha256": "6b86b273ff34fce19d6b804eff5a3f5747ada4eaa22f1d49c01e52ddb7875b4b"
},
{
"id": 2,
"server_ip": "161.47.7.14:80",
"url": "/events4mem.php?version=GUI1539&SessionID=b60b1200-2d14-4923-b640-40905c1b6d11&MinorSessionID=a28a5d9c3e46413187e4289deacid=INST",
"short_url": "/events4mem...trackutil=",
"req_head": "GET /events4mem.php?version=GUI1539&SessionID=b60b1200-2d14-4923-b640-40905c1b6d11&MinorSessionID=a28a5d9c3e46413187e4289deacid=INST HTTP/1.1",
"req_body": "ok",
"res_base64": "b2s=",
"respeek": "ok",
"magic_name": "Inconclusive. Probably text",
"magic_ext": "TEXT",
"res_head": "HTTP/1.1 200 OK\r\nServer: Apache/2.2.15 (CentOS)\r\nVary: Accept-Encoding\r\nContent-Type: text/html; charset=UTF-8\r\nDate: Thu, 11 Nov 2010 04:45:03 GMT",
"res_num": "200 OK",
"res_type": "text/html",
"host": "www.reimageplus.com",
"referrer": "",
"filename": "events4mem.php",
"method": "GET",
"epochtime": "11/28/77 04:45:03",
"res_len": "2.0 B",
"md5": "444bcb3a3fcf8389296c49467f27e1d6",
"sha256": "2689367b205c16ce32ed4200942b8b1e262dfc70d9bc9fbc77c49699a4f1df"
```

Figure B.1: Data Analysis Result

B.1.1 Feature Extraction Code

```
package com.mkkyong.seo;

import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.DocumentBuilder;
import org.w3c.dom.Document;
import org.w3c.dom.NodeList;
import org.w3c.dom.Node;
import org.w3c.dom.Element;
import java.io.File;

public class ReadXMLFile {

    public static void main(String argv[]) {

        try {

            File fXmlFile = new File("/Users/Data/IOC.xml");
            DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
            DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
            Document doc = dBuilder.parse(fXmlFile);
```

```

doc.getDocumentElement().normalize();

System.out.println("Root element :" + doc.getDocumentElement().getNodeName());

NodeList nList = doc.getElementsByTagName("IOCRule");

System.out.println("-----");

for (int temp = 0; temp < nList.getLength(); temp++) {

Node nNode = nList.item(temp);

System.out.println("\nCurrent Element :" + nNode.getNodeName());

if (nNode.getNodeType() == Node.ELEMENT_NODE) {

Element eElement = (Element) nNode;

System.out.println("ID : " + eElement.getAttribute("ID"));
System.out.println("Domain : " + eElement.getElementsByTagName("dns")
.item(0).getTextContent());
System.out.println("Classtype : " + eElement.getElementsByTagName("classtype")
.item(0).getTextContent());
System.out.println("SHA1 : " + eElement.getElementsByTagName("sha1").item(0)
.getTextContent());
System.out.println("SHA3 : " + eElement.getElementsByTagName("sha3").item(0)
.getTextContent());
System.out.println("MD5 : " + eElement.getElementsByTagName("md5").item(0)
.getTextContent());
System.out.println("Ssdeep : " + eElement.getElementsByTagName("Ssdeep")
.item(0).getTextContent());
System.out.println("Reference : " + eElement.getElementsByTagName("Reference")
.item(0).getTextContent());

}
}
} catch (Exception e) {
e.printStackTrace();
}
}
}

```

B.1.2 IOC Clustering

B.1.2.1 K-Means Clustering

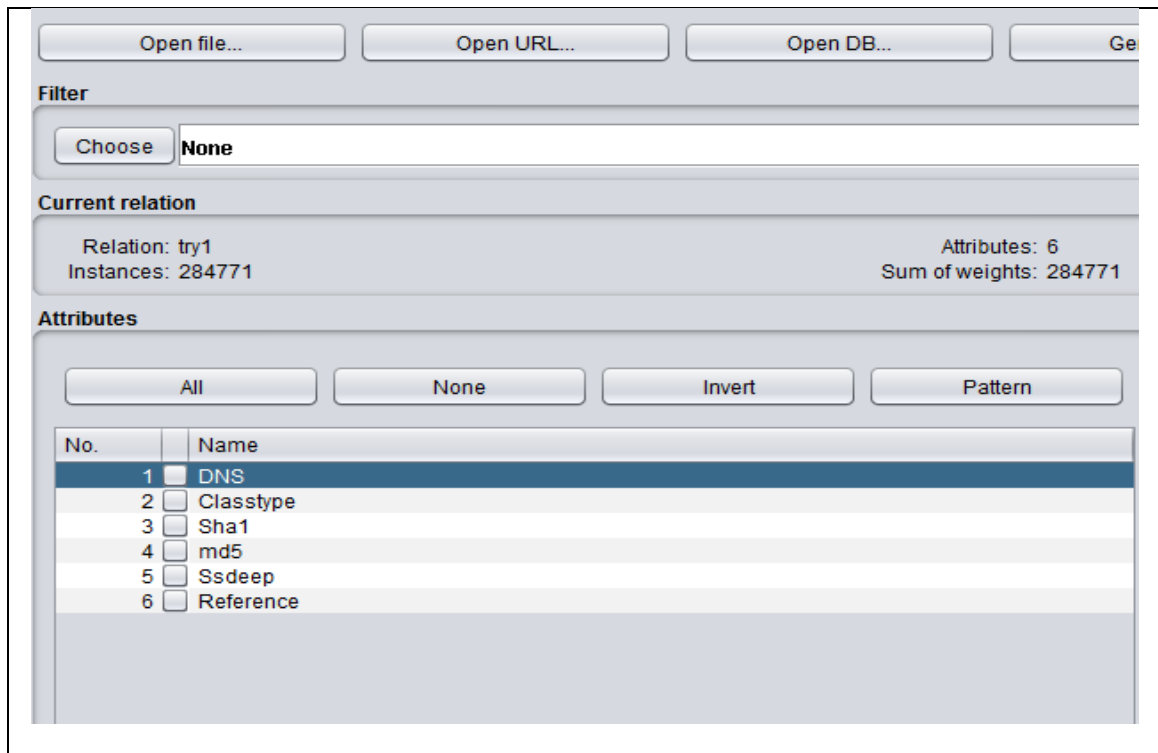


Figure 1: Weka Interface (a)

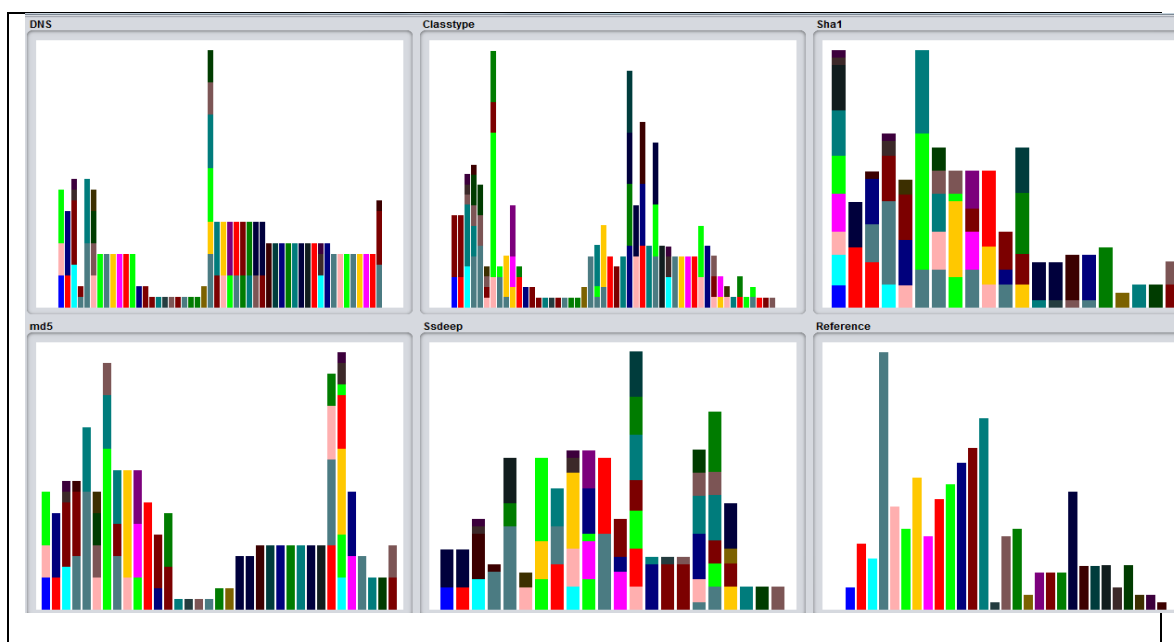


Figure 1: Weka Interface (b)

B.1.2.2 K-means Clustering Result

=== Run information ===

Scheme: weka.clusterers.SimpleKMeans -init 0 -max-candidates 100
-periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 70 -A
"weka.core.EuclideanDistance -R first-last" -I 500 -num-slots 1 -S 10

Relation: try1

Instances: 284771

Attributes: 6

DNS

Classtype

Sha1

md5

Ssdeep

Reference

Test mode: evaluate on training data

=== Clustering model (full training set) ===

kMeans

=====

Number of iterations: 2

Within cluster sum of squared errors: 111272.0

Initial starting points (random):

Cluster 0: js.himediads.com,api.ipify.org,7351ae5ad8856ff3f6721c79799a

50bcc73340af,

0580be944fdb0ca958cee222ce2c33ef,6144:+6V2lnHHnHrLrLrLrDZTGik2mtQNjV
S2

OHehYVZAwX8a5Y9TyL4xkL5:+6Vomt4jDKe6VTX8aSuIG,http://www.vapor.com

Cluster 1: obsp.verisign.com,Win32/Sality.NBA,562c87ba14fc74f230cd8f94b9

Cluster 4: i-50.b-000.xyz.bench.utorrent.com,Ransom.TeslaCrypt.

WR4,a99b6ed07c5a080059140a767f5fead9d16a4846,1dce0c0e9d07a87f56c4

35142b5994d2,6144:+6V2lnHHnHrLrLrLrLrDZ

TGiK2mtQNjVS2OHehYVZAwX8a5Y9TyL4xkL5:+6Vomt4jDKe6VTX8aSuIG,https

://www.virustotal.com/file/e8c63b3753504ff27cb93555c62bea

aded5f3e485017d950f0aa0831a9c45ac8/analysis/1445258148/

Cluster 5: b.scorecardresearch.com,PWS-Zbot.gen.ds,eda7667e108139

7ba43ccc5d6ba6d76434e05403,76cb79d267cee4b26780997f14042d1a,12288

:pwNRis33CeyBzqqW3WI94Lz2ZjEnQIV1ufX:pwt3CzzW9GGj0Q6,

https://10.0.0.40:3004/

Sha1

240c550fd056030bb171b03448c4990bf6f67f15

e0513664515eacc65e9530afe665619f2bce3802

```
fe05073567466b5388bc7f375d65bd0651513539
562c87ba14fc74f230cd8f94b9821752d210a539
f75166ac6fe6a4e191dff9196de726b0ac84b26c
md5
af575b311dd0774d393436d4a9563883
eaf85db9898d3c9101fd5fcfa4ac80e4
1dce0c0e9d07a87f56c435142b5994d2
4adfc91f1cc5545b6903a300d11dd3b0
34db7f97e0856941ed9c35716700d2a6
e23dd9cb0a032d068504d39ec5cb333a
```

Time taken to build model (full training data) : 8.86 seconds

B.1.2.3 Hierarchical Clustering

=== Run information ===

```
Scheme: weka.clusterers.HierarchicalClusterer -N 40 -L
AVERAGE -P -A "weka.core.EuclideanDistance -R first-last"
Relation: IOC2-weka.filters.unsupervised.attribute.
NominalToString-Clast
Instances: 284771
Attributes: 6
    DNS
    Classtype
    Sha1
    md5
    Ssdeep
    Reference
Test mode: evaluate on training data
```

=== Clustering model (full training set) ===

Cluster 1

```
((http://www.forensickb.com/2013/03/file-entropy-explained.html
:0,http://www.forensickb.com/2013/03/file-entropy-explained.html
:0):0,(http://www.forensickb.com/2013/03/file-entropy-explained.html
:0,http://www.forensickb.com/2013/03/file-entropy-explained.html
:0):0):1.73205,(((http://www.forensickb.com/2013/03/file-entrop
y-explained.html, :0,http://www.forensickb.com/2013/03/file-entrop
y-explained.html, :0):0,(http://www.forensickb.com/2013/03/file-
entropy-explained.html, :0,http://www.forensickb.com/2013/03/file-
entropy-explained.html, :0):0):0,http://www.forensickb.com/2013/03/f
ile-entropy-explained.html, :0):0,http://www.forensickb.com/2013/03/
```

file-entropy-explained.html, :0):0,http://www.forensickb.com/2013/03/
file-entropy-explained.html, :0):1.73205)

Cluster 26

((((http://www.forensickb.com/2013/03/file-entropy-explained.html,
:0,(http://www.forensickb.com/2013/03/file-entropy-explained.html,
:0,http://www.forensickb.com/2013/03/file-entropy-explained.html,
:0):0):0,http://www.forensickb.com/2013/03/file-entropy-explained.html,
:0):0,http://www.forensickb.com/2013/03/file-entropy-explained.html,

Time taken to build model (full training data) : 12.16 seconds

B.1.3 IOC and IOCRule

B.1.3.1 IOC with OpenIOC

```
<?xml version="1.0" encoding="us-ascii"?>
<ioc xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" id="dea18e18-3421
-45f5-b3fa-0c602f7d1e22"
last-modified="2017-6-02T23:02:53" xmlns=
"http://schemas.mandiant.com/2010/ioc">
  <short_description>Thesis IOC Logic</short_description>
  <description>This is my thesis IOC generation. </description>
  <authored_by>Ikenna Chiadighikaobi</authored_by>
  <authored_date>2017-04-07T07:45:23</authored_date>
  <links />
  <definition>
    <Indicator id="913a13b9-7b27-4545-82a7-5f79379887c4" operator="OR">
      <IndicatorItem id="1817f443-6ef0-430a-ae3b-ec67625bdf8c" condition="is">
        <Context document="FileItem" search="FileItem/Md5sum" type="mir"/>
        <Content type="md5">5a0c4e1925c76a959ab0588f683ab437</Content>
      </IndicatorItem>
      <IndicatorItem id="fa516ed8-abad-4d18-9a58-dad86b8ec673" condition="is">
        <Context document="FileItem" search="FileItem/Md5sum" type="mir"/>
        <Content type="md5">492a839a3bf9c61b7065589a18c5aa8d</Content>
      </IndicatorItem>
      <IndicatorItem id="c987c697-71b1-4618-b8b3-e4b4479fb955" condition="is">
        <Context document="FileItem" search="FileItem/Md5sum" type="mir"/>
        <Content type="md5">744a17a3bc6dbd535f568ef1e87d8b9a</Content>
      </IndicatorItem>
      <IndicatorItem id="874eaa02-c369-4e7f-bd8e-51bb3cb4b14a" condition="is">
        <Context document="FileItem" search="FileItem/Md5sum" type="mir"/>
        <Content type="md5">5c08957f05377004376e6a622406f9aa</Content>
      </IndicatorItem>
    </Indicator>
  </definition>
```

```

<IndicatorItem id="3442cfc6-7a16-4df7-bc37-2cd53628c64c" condition="is">
  <Context document="FileItem" search="FileItem/Md5sum" type="mir"/>
  <Content type="md5">8849538ef1c3471640230605c2623c67</Content>
</IndicatorItem>
<IndicatorItem id="80a94416-f08a-4428-8807-f0f8b1975be0" condition="
  contains">
  <Context document="DnsEntryItem" search="DnsEntryItem/RecordName"
    type="mir"/>

```

B.1.3.2 IOCRule for NIDS

```

#-----
# INDICATOR-COMPROMISE RULES
#-----

# alert tcp $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS
(msg:"INDICATOR-COMPROMISE
.com- potentially malicious hostname"; flow:to_server,established; content:"Host:";
http_header; content:".com-"; within:70; fast_pattern; http_header; content:!"www";
within:3; distance:-8; http_header; pcre:"/Host: [^\x0d\x0a]*?\com-/Hi";
metadata:policy security-ips drop, service http; classtype:bad-unknown; sid:32488;
rev:6;)
# alert tcp $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS
(msg:"INDICATOR-COMPROMISE
c99shell.php command request - sql"; flow:to_server,established; content:"act=sql";
fast_pattern:only; http_uri; metadata:service http; reference:url,vil.nai.com/vil/
content/v_136948.htm; classtype:policy-violation; sid:16622; rev:5;)
# alert tcp $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS
(msg:"INDICATOR-COMPROMISE
c99shell.php command request - about"; flow:to_server,established; content:"act=a
bout"; fast_pattern:only; http_uri; metadata:service http;
reference:url,vil.nai.com/vil/content/v_136948.htm; classtype:policy-violation;
sid:16616;
rev:5;)

```

Appendix C: Publication

Ikenna, R .C. (2017). Malicious Code Intrusion Detection using Machine Learning And Indicators of Compromise. *International Journal of Computer Science and Information Security*, 15(9), 160-171.



UNIMAS Graduate School
FINAL SUBMISSION OF THESIS

This form is to be filled and submitted on final submission of the thesis upon correction (as advised by the Faculty) for hardcover binding together with a copy of CD to the UNIMAS Graduate School.

Name and ID of Candidate	CHIADIGHIKA OBI IKENNA RENE 1602003				
Degree	Master*	✓	PhD*		
Programme	Master of Science				
Title of Thesis	A Dynamic Approach using Indicators of Compromise to Detect Malicious Code				
Faculty/Institute/Centre	Faculty of Computer Science and Information Technology				

*Please tick () accordingly

I hereby submit a total of 4 copies of the corrected version of my thesis for the final hardcover binding together with a copy of CD for the following distribution:-

- (i) Centre for Academic Information Services (CAIS) (MUST submitted together with a copy of CD)
- (ii) Dean/ Director of Faculty/ Institute/ Centre
- (iii) Personal
- (iv) Supervisor

Signature of the Candidate		Date

Acknowledgment of receipt of the Above <i>for</i> UNIMAS Graduate School		Date
Signature & Official Stamp		

Number of copies received	
---------------------------	--

* Please attached together a form of “Declaration of Original Work” when submitted to UNIMAS Graduate School.



GRADUATION FORM
Centre for Graduate Studies
UNIVERSITI MALAYSIA SARAWAK

Students are required to fill in this form with the correct details to facilitate the award of their degree and certificate.

Title of Thesis	A Dynamic Approach using Indicators of Compromise to Detect Malicious Code
Programme	Master of Science
Field of Study	Network Security
Faculty/ Institute	Computer Science and Information Technology
Identity Card or Passport	A05771263
Degree	Masters

Please complete your full name in the boxes provided (each alphabet in a box) as in your Identity Card or Passport.

C	H	I	A	D	I	G	H	I	K	A	O	B	I						
I	K	E	N	N	A			R	E	N	E								

(In Block Letter)

Correspondence Address (Please inform us immediately of any changes.)

14	C	H	I	E	F		O	B	I	D	I	K	E		A	Y	A		
U	M	U	E	Z	E		O	S	I	S	I	O	M	A		A	B	A	
A	B	I	A		S	T	A	T	E		N	I	G	E	R	I	A		

Signature of the Candidate		Date

Acknowledgment Signature on behalf of the Centre for Graduate Studies		Date

