

**A MODEL FOR DETECTING INFORMATION TECHNOLOGY
INFRASTRUCTURE POLICY VIOLATIONS IN A CLOUD ENVIRONMENT**

RUTH ANYANGO OGINGA

**A Thesis Report Presented to the Institute of Postgraduate Studies of Kabarak
University in Partial Fulfilment of the Requirement for the Award of Doctor of
Philosophy in Computer Science**

KABARAK UNIVERSITY

NOVEMBER 2019

DECLARATION

I, the undersigned, declare that this thesis is my original work and it has not been presented in any other university or institution for academic credit.

NAME: Ruth Anyango Oginga

REG: DGI/M/1272/09/15

Signature Date

RECOMMENDATION

To the Institute of Postgraduate studies:

This thesis entitled “**A model for Detecting Information Technology Infrastructure Policy Violations in a Cloud**” and written by **Ruth Anyango Oginga** is presented to the Institute of Postgraduate Studies of Kabarak University. We have reviewed the thesis and recommend it be accepted in partial fulfillment of the requirement for the award of Doctor of Philosophy in Computer Science.

Prof. Felix Musau

Department of Computer Science

Riara University

Signed Date

Dr. Christopher Maghanga

Department of Computer Science and IT

Kabarak University

Signed Date

COPYRIGHT

@2019

Ruth Anyango Oginga

All rights reserved. No part of this thesis may be reproduced or transmitted in any form or by any means of mechanical, including photocopying, recording or any other information storage or retrieval system without permission in writing from the author or Kabarak University.

ACKNOWLEDGMENTS

My heartfelt sincere gratitude is to the Almighty God for His grace and providing me with good health, sound mind and the knowledge to complete this thesis. I do not take it for granted. I also thank Kabarak University for giving me an opportunity to pursue my dream by availing its resources. I greatly wish to acknowledge the support of my supervisors Prof. Musau Felix and Dr. Christopher Maghanga for their wisdom and support without which I could not have come this far with my thesis. I particularly thank my loving husband Wyclife for his moral and financial support, and to our children Neziah and Ashriel for their love. To all my colleagues who contributed in one way or another in quenching my desire for knowledge, especially Mr. Ragama, Dr. Masese, Dr. Rugiri, and Dr. Karie, I am grateful. Finally, I acknowledge my extended family members for their unfailing love, encouragement and moral support throughout my period of study and for understanding and appreciating the demands of this course in terms of time and resources.

DEDICATION

This thesis is earnestly dedicated to my lovely family for their support.

ABSTRACT

The pervasiveness of the internet and available connectivity solutions brought about by cloud computing has led to an unprecedented increase in technologies built based on information technology infrastructures. This has improved the number of cloud users and substantially increasing the number of incidents related to the security of infrastructure and data in the recent past. Most organizations consider the deployment of different types of protection systems to curb various malicious activities. Organizations offer sophisticated monitoring and reporting capabilities to identify attacks against the cloud environment. Users with ill intentions have increasingly used the cloud as an attack vector due to its ubiquity, scalability and open nature despite the existence of policy violation detection systems necessitating the need to strengthen access policies from time to time. Policy violation detection plays a major role in information security by providing a systematic way of detection and interpreting attacks. Some of the known weaknesses of most detection tools are the generation of false positives or false alerts and the inability to perform analysis if traffic is encrypted as well as failure to detect and prevent attacks. This research was therefore concerned with the investigation of weaknesses of firewall and Intrusion Detection Systems (IDS) which are supported by the cloud. The information was then used to build and experiment on an improved model of a policy violation detection system. Experiments revealed the weakness in existing systems specifically IDS and firewalls. Unlike the existing systems, a new model designed to overcome the shortfall was able to detect both recognized and unrecognized attacks and signatures. Moreover, the model is capable of preventing the occurrence of false positives and terminates suspicious nodes in real time without human intervention. An additional area of application such as movement from data from one cloud to another is not achievable, because of the mixed environment of the cloud. This is a potential area for investigation in the future.

Keywords: Policy violation, Develop, Cloud, Detection, Weaknesses, Attacks

TABLE OF CONTENT

DECLARATION.....	ii
RECOMMENDATION.....	iii
COPYRIGHT.....	iv
ACKNOWLEDGMENTS.....	v
DEDICATION.....	vi
ABSTRACT.....	vii
TABLE OF CONTENT.....	viii
LIST OF FIGURES.....	xiii
LIST OF TABLES.....	xvi
LIST OF ABBREVIATION AND ACRONYMS.....	xvii
OPERATIONAL DEFINITION OF TERMS.....	xix
CHAPTER ONE.....	1
INTRODUCTION.....	1
1.1 Introduction.....	1
1.2 Background of the Study.....	1
1.2.1 Detection systems.....	3
1.3 Statement of the Problem.....	5
1.4 Purpose of the Study.....	6
1.4.1 Objectives of the Study.....	6
1.4.2 Research questions.....	6
1.5 Justification of the study.....	7
1.6 Scope of the Study.....	8
1.7 Limitations of the Study.....	8
1.8 Assumptions of the Study.....	8
CHAPTER TWO.....	10
LITERATURE REVIEW.....	10
2.1 Introduction.....	10
2.2 Overview of Cloud Environment.....	10
2.2.1 Characteristics of the Cloud Computing.....	12
2.2.2 Cloud Computing Service Models.....	13
2.2.3 Cloud Delivery Models.....	14
2.2.3.4 Community Cloud.....	16
2.2.4 Detection tools.....	16

2.2.4.1 Intrusion Detection System	18
2.2.5 Intrusion Prevention System	24
2.2.6 Firewall.....	27
2.2.7 Types of Firewall.....	28
2.2.8 Flow Chart of a Firewall.....	29
2.3 Weaknesses of Existing Detection Tools	30
2.3.1 Weaknesses of Intrusion Detection System.....	31
2.3.2 Weaknesses of the Intrusion Prevention System	32
2.3.3 Weaknesses of Firewall	32
2.4 Demonstrate the weaknesses of existing detection tools on policy violation in the cloud	34
2.5 Develop a model to detect and identify policy violation in the real-time traffic	40
2.6 Cloud computing crimes	44
2.6.1 Virtual Machine Attacks.....	45
2.6.2 U2R (User to Root Attacks)	46
2.6.3 Insider Attacks	46
2.6.4 Denial of Service (DOS) Attack.....	47
2.6.5 Port Scanning.....	47
2.6.6 Backdoor Path Attacks	48
2.6.7 User Spoofing	48
2.6.8 Penetration Attack	49
2.6.9 Malware Injection Attacks.....	50
2.6.10 Cross VM Side-Channel Attacks.....	51
2.6.11 Theft of Service Attacks	51
2.7 Review of Models Developed	53
2.8 Experimental Model on Curbing the Weaknesses of IDS and Firewall on Policy Violation in Cloud.....	59
2.9 Cloud Security Policy.....	61
2.9.1 Security Goals for Cloud Computing	63
2.10 Hail marry Attack in Armitage.....	64
2.11 Software Development Models	64
2.11.1 Waterfall Model.....	64
2.11.2 Rapid Application Development	66
2.12 Gaps Identified from Previous Studies.....	68

2.13 Conceptual Framework	69
CHAPTER THREE.....	71
RESEARCH DESIGN AND METHODOLOGY	71
3.1 Introduction	71
3.2 Research Design.....	71
3.3 Identification of the Weaknesses of IDS and Firewall.....	72
3.4 Prototype Development.....	73
3.5 PROVIDE Model Development	76
3.6 Network Diagram.....	77
3.7 Population of the Study	78
3.8 Data Analysis	79
3.9 Reliability and Validity of the Instrument.....	80
3.10 Model Evaluation	80
3.11 Research Authorization	81
3.12 Ethical Considerations.....	81
CHAPTER FOUR.....	83
DATA ANALYSIS, PRESENTATION, AND DISCUSSIONS	83
4.1 Introduction	83
4.2 Analysis.....	83
4.3 Weaknesses of IDS and Firewall	84
4.3.1. Demonstration of the existing Weaknesses of the Firewall	85
4.3.2. Experimental Setup for the Firewall	85
4.3.3Algorithm for Demonstrating Weaknesses of Firewall	86
4.3.6. Login Credentials	90
4.3.7 Penetration Stage of Firewall	93
4.3.8 Scanning Stage of Firewall.....	99
4.4 Weaknesses of Intrusion Detection System	102
4.4.1 Demonstration of the existing Weaknesses of the Intrusion Detection system.....	102
4.4.2 Experimental set up of IDS	103
4.4.3 Algorithm for Demonstrating Weaknesses of IDS	103
4.4.4 Flowchart Diagram for IDS	104
4.4.5 Snort Configuration	107
4.4.6 False Positive	108

4.4.7 False negative	115
4.5 Design of POVIDE Model	116
4.5.1 Flowchart Diagram of POVIDE Model.....	116
4.5.2 Policy Violation Detection Model Architecture	119
4.5.3 POVIDE Model Algorithm	121
4.5.4 POVIDE Model Experimental Setup	121
4.5.5 Configuration Stage	123
4.5.6 Scanning Stage	124
4.5.7 Penetration Stage	127
4.5.8 Analysis of Logs	130
4.6 Proof of Concept	132
4.7 Model Quick Design and Prototype Building	133
4.7.1 Requirement Gathering.....	134
4.7.2 Use Case Diagram for POVIDE Model	137
4.7.3 Sequence Diagram for POVIDE Model	138
4.7.4 Class Diagram.....	139
4.7.5 Component Diagram.....	141
4.8 Model Development.....	142
4.9 POVIDE Model Testing.....	143
4. 10 High-Level Overview of the POVIDE Model	145
4.10.1 Virtualization Tier	146
4.10.2 Application Tier.....	146
4.10.3 File System Tier.....	146
4.10.4 Policy Tier	146
4.11 Experiment Purpose and Scenarios Used.....	147
4.12 Execution of the Model	149
4.13 Evaluation.....	150
4.13.1 Criteria.....	150
4.14 Comparison of Existing Detection Systems Versus POVIDE Model.....	154
CHAPTER FIVE	155
SUMMARY, CONCLUSION, AND RECOMMENDATIONS.....	155
5.1 Introduction.....	155
5.2 Summary	155

5.3 Conclusions	156
5.4 Recommendations	159
5.4.1 Policy recommendations.....	160
5.4.2 Recommendation for Future work.....	161
REFERENCE	163
APPENDICES	180
Appendix I: Evaluation Form.....	180
Appendix II: Letter of Introduction.....	181
Appendix III: Authorization letter from NACOSTI	182
Appendix IV: Permit from NACOSTI.....	183
Appendix V: Sample code.....	184

LIST OF FIGURES

Figure 1: Classification of Detection tools	17
Figure 2: Intrusion Detection System	19
Figure 3: Snort IDS	22
Figure 4: Intrusion Prevention System	26
Figure 5: Firewall	27
Figure 6: Flowchart of a firewall	30
Figure 7: FlowGuard framework	42
Figure 8: Virtualized Computing	42
Figure 9: Threat Model	44
Figure 10: Ucloud Architecture	59
Figure 11: IDS using CAPTCHA as a trap	60
Figure 12: Conceptual Framework	70
Figure 13: Rapid Application Development.....	72
Figure 14: Rapid Prototyping	74
Figure 15: Network Diagram	78
Figure 16: Firewall Setup	86
Figure 17: Flowchart Diagram for Firewall	89
Figure 18: Physical Volume Usage	90
Figure 19: Login Credentials	91
Figure 20: Incorrect Logging	91
Figure 21: Connection to the network	92
Figure 22: Penetration of Exploit	94
Figure 23: Exploited File	96
Figure 24: Exploited Virtual Machine	97
Figure 25: Saved Captured file	97
Figure 26: Screenshot File	98
Figure 27: Scanning by Internal Network.....	100
Figure 28: Tools used for Scanning.....	101
Figure 29: Zenmap.....	102
Figure 30: IDS Setup	103
Figure 31: Flowchart Diagram for IDS	106

Figure 32: Snort Configuration	107
Figure 33: Log File	108
Figure 34: False Alert	109
Figure 35: Start Apache2	110
Figure 36: Exploiting Tools	111
Figure 37: Metasploit	112
Figure 38: Launching an Exploit	113
Figure 39: Downloaded	113
Figure 40: Virtual Machine versus the attackers' Machine	114
Figure 41: IDS Router Connection	115
Figure 42: Flowchart Diagram of PROVIDE Model	118
Figure 43: Policy Violation Detection Model Architecture	120
Figure 44: PROVIDE Model Setup	122
Figure 45: Configuration Stage.....	123
Figure 46: Scanning	125
Figure 47: Allowed and Denied Ports	126
Figure 48: Normal Site	127
Figure 49: False positive	128
Figure 50: Exploited Application	129
Figure 51: False negative	130
Figure 52: Analysis of logs	131
Figure 53: Analysis of tcpdup.log	132
Figure 54: Attack Responses Rules	132
Figure 55: Model Quick Design	133
Figure 56: Requirement Gathering	134
Figure 57: Functional Requirement	135
Figure 58: Non- Functional Requirements for PROVIDE Model	136
Figure 59: Use Case Diagram of PROVIDE Model	137
Figure 60: Sequence Diagram of PROVIDE Model	138
Figure 61: Class Diagram	140
Figure 62: Component Diagram	142
Figure 63: PROVIDE Model Interface	143
Figure 64: List of Logins	145
Figure 65: High-level Overview of the PROVIDE Model	147

Figure 66: POVIDE Model Execution..... 149

LIST OF TABLES

Table 1: Types of Intrusion Prevention System	26
Table 2: Policy and Baseline Controls	62
Table 3: Strength and Weaknesses of Sdlc Models	67
Table 4: Organizations and Institutions	79
Table 5: Reliability and Validity Test Results	80
Table 6: Evaluation Procedure	81
Table 7: Weaknesses of IDS, IPS, and Firewall	85
Table 8: Criteria for Assessing Goodness of Fit	144
Table 9: Analysis of Parameter Estimates	144
Table 10: Evaluation Criteria	151
Table 11: Comparison of Existing Detection System Versus Povid Model	154

LIST OF ABBREVIATION AND ACRONYMS

Abbreviations	Meaning
AC	Access Control
AD	Anomaly Detection
ADS	Anomaly Detection System
AUP	Acceptable Use Policy
AWS	Amazon Web Service
CSA	Cloud Security Alliance
CSP	Cloud Service Provider
DAC	Discretionary Access Control
DIDS	Distributed Intrusion Detection Systems
Desvi	Detection SLA violation infrastructure
DMZ	Demilitarized Zone
DoS	Denial of Service
DDoS	Distributed Denial of Service
ERP	Enterprise Resource Planning
FIM	Federated Identity Manager
GCCIDS	Grid and Cloud Computing Intrusion Detection System
SaaS	Infrastructure as a Service
IDM	Internet Download Manager
IDP	Intrusion Detection and Prevention
IDS	Intrusion Detection System
IPS	Intrusion Prevention System
KVM	Kernel-based Virtual Machine
NETM	Network Monitoring Application

NIDS	Network Intrusion Detection System
PaaS	Platform as a Service
POVIDE	Policy Violation Detection Model
QoS	Quality of Service
SaaS	Software as a Service
SD	Signature Detection
SDKs	Software Development Kits
SLA	Service Level Agreement
SMVR	Secure Management Techniques for Virtualized Resources
SSLA	Security Service Level Agreement
StegAD	Steganography Active Defence
SYN	Synchronize
TCP	Transmission Control Protocol
UAP	User Acceptable Policy
UML	Unified Modelling Language
UTM	Unified Threat Management
VCL	Virtual Computing Laboratory
VESPA	Virtual Environment-based Self Protecting Architecture
VM	Virtual Machines
VMM	Virtual Machine Manager
VPN	Virtual Private Network

OPERATIONAL DEFINITION OF TERMS

The attack	is an attempt to actively exploit weaknesses in a cloud environment.
Model	is a program that is designed to detect and prevent intrusions in real time.
Virtual machine (VM):	is an emulation of a physical computer system. It provides the functionality of a normal working physical computer system.
Cloud computing:	means storing and accessing data and programs on the Internet as an alternative to a computer.
Detect:	is to discover or identify the presence or existence of a violation
Intrusion detection	is the identification of the actions occurring in a computer system or network and analyzing them for signs of possible concern, which are violations to computer security policies or threats
IT Infrastructure:	as a combined set of hardware, software, networks, facilities, including all of the information technology-related equipment, used to develop, test, deliver, monitor, control or support IT services

- Policy Violation** is a certain reason or event of discharge that is said to be unlawful, and which violate the guidelines or policies laid down by the model.
- Resource pooling:** Physical computer systems and virtual resources that are not used are dynamically assigned and reassigned according to the demand for use.
- Sandbox:** is the facility to examine the host system that is prohibited or restricted. In this sense, sandboxes are a detailed example of virtualization.
- Virtual Box** is open source software used to hold POVIDE Model
- Cloud Environment:** is using a web-based application for every task rather than installing software or storing data on a computer

CHAPTER ONE

INTRODUCTION

1.1 Introduction

This chapter gives a brief background of the main concepts and problems informing this study. It further proceeds to state the research problem, outlines the research objectives and research questions. It also provides the justification for carrying out the research in the selected area.

1.2 Background of the Study

The number of business organizations moving towards the cloud is increasing very rapidly. The ease of use and the connectivity the cloud provides are highly useful but the risks involved and malicious intrusions are also increasing day by day. Most organizations consider the deployment of different types of protection systems to curb various malicious activities. The exploitation of computer networks is getting more common. It is very serious for a business organization as well as users to guard their data from severe threats that would aim to steal or interfere with their information.

According to Stanoevska-Slabeva & Wozniak (2010), the cloud is the services or demand resources over the Internet scale. Cloud computing is a style of computing in which dynamically scalable and often virtualized resources are provided as a service over the Internet. Cloud Computing has transformed many organizations in several ways. First, organizations can consolidate the infrastructure, since the deployment of virtual machines can replace the use of physical machines. Since there are fewer computers, people and spaces being used, these help organizations reduce the operational costs in the long-term (Ramachandran et al., 2015).

The cloud has become an increasingly popular service due to the numerous advantages it possesses. These include high computing power, cheaper cost of services, better performance, scalability, and accessibility among others (Hashem et al., 2015). It dramatically lowers the cost of entry for small firms trying to benefit from compute-intensive business analytics that was previously available only to large corporations. It makes it easier for enterprises to scale their services, which are increasingly reliant on accurate information according to client demand. Cloud also makes possible new classes of applications and delivers services that were not possible before (Avram, 2014). It has revolutionized the IT world with the provisioning of its services infrastructure, less maintenance cost, data, and services availability assurance, rapid accessibility, and scalability (Garg et al., 2013).

The service has three basic abstraction layers for example system layer, which is a virtual machine abstraction of a server, platform layer for the virtualized operating system of a server, and application layer that includes web applications (Rosenblum & Garfinkel, 2005). It also has three service models which are Platform as a Service (PaaS), Infrastructure as a Service (IaaS) and Software as a Service (SaaS) models. PaaS model facilitates users by providing a platform on which applications can be developed and run. IaaS deliver services to users by maintaining large infrastructures like hosting servers, managing networks and other resources for clients. SaaS model makes user worry-free of installing and running software services on its own machines (Subashini, & Kavitha, 2011)

As cloud technology becomes immensely popular among these businesses, the question arises: Which cloud model should one consider for business? There are four types of cloud models available in the market: Public, Private, Hybrid, and Community (Goyal, 2014). However, since this kind of computing paradigm is new, it has shortfalls that need to be addressed to make cloud-computing services more convenient to use. Shortfalls include reliability, connectivity, open access, security and privacy (Stergiou et al., 2018). Security is

viewed as one of the main adoption stoppers to cloud computing and the complexity of infrastructure involved leaves the door open to various threats coming from outside and within. Intrusions, malware or security policy violations of curious or malicious users are just but a few. Control assets and information policies are required in order to protect the organizational data of the cloud-computing environment. Acceptable use policy is needed to make sure controls and monitoring of services are provided. Acceptable use policy is a set of rules applied by organizational network administrators to restrict the ways in which the network is used and set guidelines as to how it should be used.

Different network administrators use different types of network-based and host-based security software to detect malicious activities in the cloud. The main target of the assailants is to make an attack on the presented resources in the Cloud computing settings (Hameed et al., 2016).

1.2.1 Detection systems

Intrusion is the act of violating the security policy that pertains to an information system. Intrusion detection can be defined as the act of detecting actions that attempt to compromise the confidentiality, integrity or availability of a resource (Patel et al., 2010). Intrusion detection is the process of monitoring the events occurring in a computer system or network and analyzing them for signs of possible incidents, which are violations of computer security policies, acceptable use policies or standard security practices (Scarfone, & Mell, 2012). Incidents have many causes such as malware, attackers gaining unauthorized access to systems from the Internet, and authorized users of systems who misuse their privileges or attempt to gain additional privileges for which they are not authorized (Kaur et al., 2014). Thus, detection systems monitor and alert attacks in the cloud. There are different types of

detection systems used in the cloud. The most used ones are Intrusion Detection System and Firewall.

Intrusion Detection System (IDS) is a software tool or device that monitors the system or activities of the network for policy violations or malicious activities and generates reports to the network administrator. IDS may be used to monitor a network from outside threats, like monitoring a network for problems caused by such events as running an application that accesses sites or downloads software that is not allowed, and malicious network activities (Kaur et al., 2014). The system collects and analyzes data from a range of areas within a computer or a network to identify possible security violations. The intrusions may include attacks both from outside the organization as well as within the organization. Different methods can be used to identify intrusions but each one is specific to a specific method. The most important goal of an intrusion detection system is to detect the attacks proficiently. Furthermore, it is equally important to detect attacks at the beginning stage in order to reduce their impacts (Jabez, & Muthukumar, 2015). However, the current cloud IDS fails short of the practical capability to prevent attacks at its initial stage. Thus, Intrusion Prevention System (IPS) is preferred over IDS in order to automatically take action towards suspected network activities. The IPS can be constructed based on IDS because the detection function is needed in an IPS solution. However, most existing IPS solutions are designed for traditional network and simple migration is not effective enough to detect and defend malicious attacks in the cloud (Bace, & Mell, 2001).

Another prevention tool is a firewall. A firewall is a combination of hardware and software that isolates an organization's internal network from other networks, allowing some packets to pass and blocking others. It functions to avoid unauthorized or illegal sessions established to the devices in the network areas it protects. Firewalls are configured to protect against unauthenticated interactive logins from the outside world. However, Firewalls cannot

prevent attacks coming from Intranet (Kaur et al., 2014). Thus, detection systems are essential in detecting intrusions. Detection and prevention tools have weaknesses that need to be solved. Equally, methods need to detect the attacks as they arise and prevent the attackers from producing threat activities inside the cloud respectively.

1.3 Statement of the Problem

Most organizations nowadays use of acceptable use policy to stipulate the events illegal to the users of an organization's IT infrastructure. Every user is required to follow all the policies specified in the acceptable use policy document. Despite the use of existing detection and prevention systems such as IDS, IPS, and Firewall to detect and prevent malicious activities and to analyze data that originates from the host computer, some users circumvent detection and prevention tools to access the cloud. The greatest challenge with most of the detection and prevention technologies is the generation of false positives or false alerts. Furthermore, existing detection tools are unable to perform analysis if traffic is encrypted in real time. Despite firewalls widely accepted, limitations make it less efficient with the current security challenges. These include the inability of the firewall to prevent attacks coming from Intranet. This limits its ability to fend off internal attacks. The firewall also uses a set of rules that are manually configured to differentiate genuine traffic from non-legitimate traffic. Additionally, firewalls cannot react to a network attack nor can it initiate effective countermeasures (Hock, & Kortis, 2015). On the other hand, in spite of having IDS and IPS as detection and prevention tools, they both have some limitations that hinder them from being able to detect and block attacks. Some of the challenges they pose include susceptibility to protocol-based attacks that mean that encrypted packets are not processed by IDS. Another challenge is that IP packets can still be faked (Chowdhary et al., 2014). Moreover, the network administrator is mandated to scrutinize the attack once it is detected and reported.

Therefore, there is a need to develop a Policy Violation Detection Model to solve the aforementioned challenges.

1.4 Purpose of the Study

The purpose of the study was to develop a model for detecting Information Technology infrastructure policy violations in a cloud environment.

1.4.1 Objectives of the Study

- i. To identify weaknesses in IDS, IPS and firewall as tools of intrusion detection and prevention
- ii. To demonstrate the weaknesses of IDS, IPS and firewall on policy violation in the cloud
- iii. To develop a model to detect and identify policy violation in real time traffic
- iv. To evaluate the performance of the model in curbing the weaknesses of IDS, IPS, and firewall on policy violation in the cloud

1.4.2 Research questions

- i. What are the weaknesses in IDS, IPS, and firewall as tools of Intrusion Detection and Prevention?
- ii. How can you demonstrate the weaknesses of IDS, IPS, and firewall on policy violation in the cloud?
- iii. How can you develop a model to detect and identify policy violations in real time traffic?
- iv. Does the performance of the model curb the weaknesses of IDS, IPS, and firewall on policy violation?

1.5 Justification of the study

The policy violation detection model stands to benefit organizations and institutions using cloud computing. The model is meant to detect hosts in real time trying to violate policies by terminating the node session that is being used for the attack on the network which is critical in this era of cloud computing.

The existing detection and prevention tools such as IDS, IPS and firewall have weaknesses of sending false alarms. The greatest challenge with the majority of detection technologies is the generation of false positives or false negatives. The policy violation detection model drastically deals with false alarms. This is because either any activity is authorized or unauthorized terming it efficient at detecting attacks.

Most detection technologies have challenges in identifying unknown attacks. This means that they can only detect known attacks. Policy Violation Detection Model identifies the unknown and known attacks in real time.

It has also been pointed out that existing detection tools are incapable of performing analysis when the traffic is encrypted. They cannot detect attacks inside a virtual network contained by the hypervisor. The model is able to identify, detect, and block threats in real time and curb internal and external attacks.

Finally yet importantly analytical module has a limited ability to analyze the source information that is collected during intrusion detection. Only a portion of the source information is buffered. There is a large volume of traffic that is not analyzed. Therefore, the network administrator will not able to tell where the threat originated. POVIDE Model is able to analyze the whole traffic. It is also able to tell where the attacks originated from through submitting the IP address of the attacker's machine.

Lastly, the implementation of the model leads to the satisfaction derived from having contributed to the development of a solution that solves the problems affecting the organizations and institutions.

1.6 Scope of the Study

Expert users evaluated the model on its ability to curb unauthorized intrusions. It was geared towards addressing the weaknesses of existing detection and prevention systems specifically IDS, IPS, and firewall.

1.7 Limitations of the Study

The experimentation of the developed POVIDE Model required at least 8GB of memory in order to work at a convenient speed. To overcome this limitation a new laptop with the required specification was purchased.

There was A firewall constraint concerning the Cloud technology used. However, the use of open source tools provided a way of opposing these concerns because some open source tools do work with Linux-based Cloud systems with modest or no alteration.

Finally, there were limitations in terms of the number of physical systems available for the experiments, where more than two systems were needed. These were replaced with alternative Virtual Machines. Therefore, an experiment was conducted using VMs so that they could be compared.

1.8 Assumptions of the Study

The development of the POVIDE Model assumed that specific experts who tested the model had the knowledge of policy violations in the cloud and that they have internet-enabled devices that can access and download data from the network. It is also assumed that the

experts who tested the PROVIDE Model gave honest and expert opinions during the experiment. This assumption was realized by making sure that the individuals were trained on the Model.

Lastly, the user's access process also assumed that the registration data would be crosschecked against the cloud service providers' or network admin.

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

The purpose of this chapter is to provide a summary of the literature on the developments in the area of policy violations in a cloud environment. It starts with an overview of what has been done by scholars in the cloud environment. Then it continues to review literature based on objectives and provide relevant literature with knowledge used to develop the model. Furthermore, the chapter gives compressive literature on cloud computing. Finally, the chapter describes the conceptual framework for this research.

2.2 Overview of Cloud Environment

As discussed earlier, cloud computing presents users with access to low-cost computing resources, powerful processing, storage, and networking. While these resources obviously provide benefits to numerous users, there are also concerns that users can easily gain access thus putting the security of the cloud into question (Zhou et al., 2010).

Numerous research works propose intrusion detection systems. The proposed systems face security challenges while in use in public and private cloud infrastructures, and involve network administrators who must understand and be concerned with network security (Jing et al., 2014). The network must be monitored given the increased risk posed by interrelated computers. According to Snapp et al. (2017), the intrusion detection system is used to monitor a network from external threats. The network monitoring system can be used to monitor the network for harm triggered by an event such as malicious applications and intrusions. It also monitors threats that can crash network servers, malicious network activities, and network connections, among others.

According to Quah and Rohm (2013), respondents believed that cloud computing made it harder for organizations to find a way to protect users' data and the biggest concern

was regarding the risk of losing control over data locations and illegal access to data. Theoharidou et al., (2013) concluded that auditors and authorities need to be able to hold service providers responsible for their actions, enforcing rules and regulations through penalties and other mechanisms, and ensuring that problems are remedied promptly and sufficiently.

According to Pearson (2013), organizations must trust their cloud service providers. Organizations must make sure that their clients also trust the same cloud service provider. They must have knowledge of the cloud. A cloud service provider and the customer often enter into a contractual relationship to establish trust. Typically, an organization may be compensated in an event that the service is not delivered as expected. However, in the modern computing world, establishing trust in cloud computing is related to preventing a trust violation rather than to compensate for a violation in case it occurs (Velte et al., 2010). For any modern organization, an irreparable security breach or compensation cannot bring back lost data or the organization's reputation.

Detecting misuse by legitimate users abusing their privileges is a complex, multi-faceted problem. This is because malicious insiders can engage in a variety of activities and use knowledge of their organization's systems and networks to avoid detection (Maloof, & Stephens, 2014). Modern malware designers and cyber attackers are innovative and constantly seeking to circumvent existing measures generating different versions of malware using alteration. Existing IDS and IPS approaches are considered to detect unauthorized access attempts and Distributed Denial of Service (DDoS) attacks (Shi et al., 2012). For example, Alsunbul et al. (2014) presented a network defense system for detecting and preventing unauthorized access attempts by dynamically generating a new protocol to replace the standard protocol. The aim is to confuse scanning attempts. The network path is also

changed periodically to prevent unauthorized access and scanning of traffic. However, the number of packets generated can be excessive.

2.2.1 Characteristics of the Cloud Computing

According to Gong et al., (2010) essential characteristics of cloud computing were first defined by the National Institute of Standards and Technology (NIST) and have subsequently been redefined by a number of scholars and experts. These characteristics are elaborated below-

2.2.1.1 Flexibility/Elasticity

The on-demand model of cloud provisioning attached with high levels of automation, virtualization, and ubiquitous, consistent and high-speed connectivity provides for the capability to rapidly expand or contract resource allocation to service definition and requirements using a self-service model that scales to as-needed capacity. Since resources are shared, better utilization and service levels can be achieved (Dash et al., 2014).

2.2.1.2 Accessible Anywhere

According to Wahlgren et al. (2013), cloud customers are capable of accessing their data and service irrespective of the physical location. Therefore, the cloud user has no control or whereabouts of the location of the assets. Likewise, cloud providers do not have restrictions over the location of its users.

2.2.1.3 Reliability

Clouds are usually copied on multiple redundant sites which make cloud computing suitable for business continuity and disaster recovery. Consequently, cloud users can be located in areas where electricity and real estate prices are lower eventually lowering their start-up and running costs (Rittinghouse, & Ransome, 2016).

2.2.2 Cloud Computing Service Models

Understanding the layer dependence of cloud service models is very dangerous to analyze the security risks of cloud computing. These service model layers are:

2.2.2.1 Infrastructure-as-a-Service

This cloud service model normally provides access to networking structures, computers virtual or on dedicated hardware, and data storage space (Sen, 2015). Users have allocated storage capacity and start, stop, access and configure the virtual servers and storage as desired. Cloud providers connect various operating systems and their application software on the cloud communications thereafter installing their applications (Tao et al., 2014). In this representation, the cloud users are accountable for maintaining the operating systems and the application software. Infrastructure-as-a-service (IaaS) cloud providers classically charge their customers on convenience computing beginning with the cost that reflects the number of resources allocated and consumed (Erl et al., 2013).

2.2.2.2 Platform-as-a-Service

In the PaaS models, the cloud users do not control the networking structures, essential software, hardware or servers. The cloud providers supply a computing platform, which

comprises the operating system (OS), programming language execution environment, the database, and web server (Oliveira et al., 2014). Consequently, the application designers and developers run their software solutions on a cloud platform without paying for expensive hardware and software layers that usually have significant costs. PaaS eliminates the need for organizations to manage the original infrastructure and allows the cloud users to focus on the operation and management of their applications (Almorsy et al., 2016).

2.2.2.3 Software-as-a-Service

SaaS offers cloud users with a complete artifact that is run and managed by cloud providers. In the SaaS model, the customer does not supervise or control the principal cloud infrastructure as well as networking structures, computers virtual or on dedicated hardware, and data storage space except the limited user-specific application configuration settings (Huang, & Wu, 2017). All these models are known to be very promising internet-based computing platforms.

Kavis (2014) concluded that these models could result in a loss of security over customer data. This frequently happens because the enterprise IT assets are hosted on third-party cloud computing platforms. Where cloud computing can help organizations achieve more by paying less and breaking the physical limitations between IT infrastructure and its users.

2.2.3 Cloud Delivery Models

There are four basic cloud delivery models, as outlined by NIST (Liu et al., 2011) based on providers of cloud services. The organizations may employ one model or a combination of different models for efficient and optimized delivery of applications and business services. These four delivery models are:

2.2.3.1 Public Cloud

According to Sen (2015), public clouds are the most used among end-users due to their speedy setup time and low cost. Service providers of this type of cloud usually partition their physical servers and lease these portions to the cloud users. Consequently, the end-users have the notion of managing a vast computational power and storage capacity. Nevertheless, public clouds suffer from a lack of infrastructure transparency which makes them less attractive to large organizations. On the other hand, Private clouds are not open to public users in the sense that their infrastructure is controlled by private organizations.

2.2.3.2 Private Cloud

A private cloud is set up within an organization's internal activity data center. It is easier to align with security, compliance, and regulatory requirements, and provides more enterprise control over deployment and use. In the private cloud, scalable resources and virtual applications provided by the cloud providers are joint together and accessible to cloud users to share and use. It varies from the public cloud in that all the cloud resources and applications are managed by the organization itself, similar to Intranet functionality. Consumption on the private cloud is much more secure than that of the public cloud because of its precise internal coverage. Only the organization and selected stakeholders may have access to operate on a specific Private cloud (Puthal et al., 2015)

2.2.3.3 Hybrid Cloud

According to Gangwar et al. (2015), a hybrid cloud combines the features of the public and private cloud. This varied environment is appropriate for organizations that have software or hardware compatibility issues but still, want to take advantage of the vast storage space and other cloud resources provided by public clouds. An additional reason to prefer hybrid clouds is its edibility in revealing the organization's assets for a limited time to the

public users. Hence, an organization's resources can be moderately exposed to the public side of the cloud rather than risking everything on the public cloud (Assuncao et al., 2015).

2.2.3.4 Community Cloud

According to Jula et al. (2014), community cloud in computing is a joint effort in which infrastructure is shared between numerous organizations. Community with common concerns are managed internally or by a third party and hosted internally or externally. This is controlled and used by a group of organizations that have a common interest. The costs are spread over fewer users than a public cloud but more than a private cloud, so only some of the cost savings potential of cloud computing are realized.

2.2.4 Detection tools

Detection Tool assists users with the detection of the security vulnerability. There are two different genres of tools in the intrusion protection world: intrusion detection systems and intrusion prevention systems as shown in Figure 1. Though they serve a different function, there is often some overlap between the two types of tools. As its name implies, the intrusion detection detects intrusion attempts and doubtful activities in general. When it happens, it classically triggers some sort of alarm or notification. It is then up to the network administrator to take the required steps to stop or block this attempt. Intrusion detection systems can be, very expensive. IPS are network security applications, that monitor and change network, and system activities if found suspicious. The main purpose of IPSs is to recognize malicious activity and attempt to block or stop that activity (Scarfone, & Mell, 2012). Since they are not in the focus of this report, it is necessary to note here, that they are important parts of network security and strongly related to intrusion detection. IPSs can be considered extensions of IDSs (Newman, 2009).

Fortunately, there are quite a few free alternatives available out there. The research has been made on the Internet for some of the common intrusion detection software tools. These are OSSEC, SURICATE, Snort, and Sagan (Taha, & Hadi, 2019).

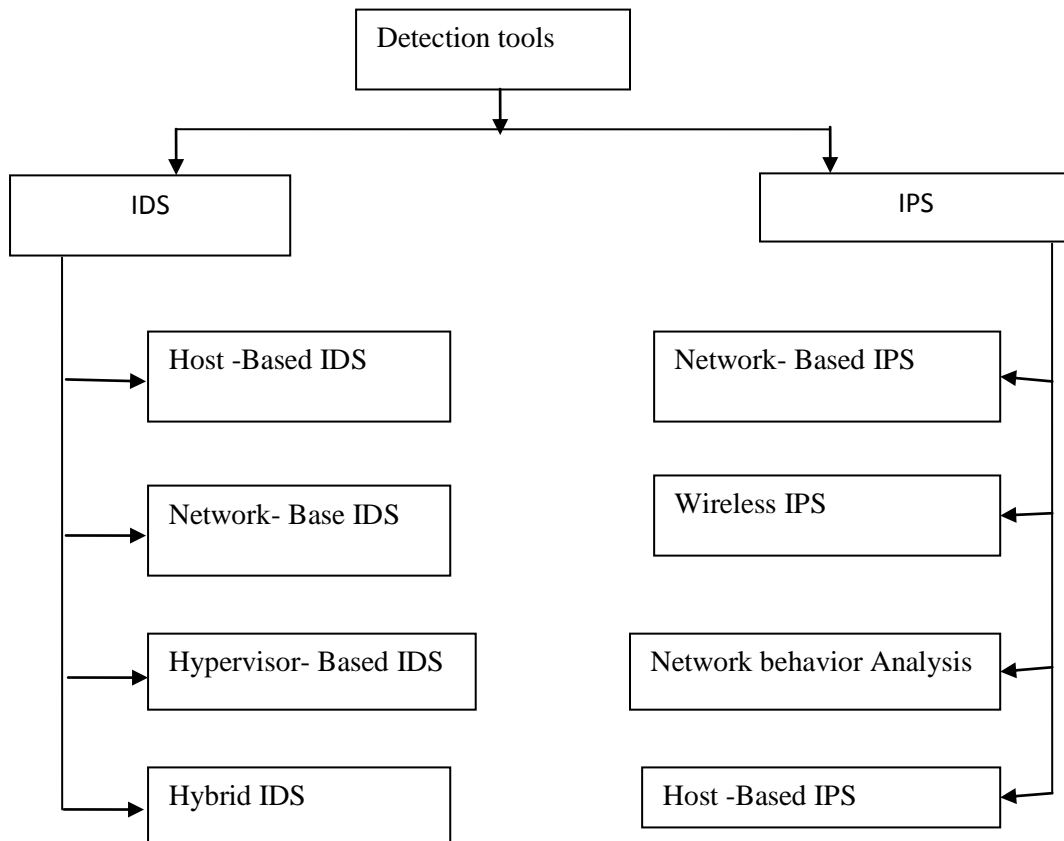


Figure 1: Classification of Detection tools

According to Buck and Hanf (2010), promising tools and techniques that can be part of an IT infrastructure to assist network administrators in detecting Acceptable Use Policy (AUP) violations are of utmost importance. Such tools and practices can further help in gathering relevant evidence data from host computers and other IT infrastructures as well. Furthermore, with the rapidly changing technological environment, it is clear that additional speedy changes in the way security incidences are detected and the way security data is analyzed needs to be done. Some of the intrusion detection discussed includes:

2.2.4.1 Intrusion Detection System

Intrusion detection systems constantly monitor a given computer network for invasion or abnormal activity. An IDS gathers and analyzes information from various areas within a computer or a network to identify possible security breaches. IDS has been used as a vital instrument in defending the network from malicious or abnormal activity. It is appropriate to know which intrusions have occurred or are happening. It is easier to understand security threats, risks and thus be better prepared for future attacks. With the ability to analyze network traffic and recognize incoming and ongoing network attacks, the majority of network administrators have turned to IDS to help them in detecting anomalies in network traffic (Chowdhary et al., 2014).

The current trend for the IDS makes it possible to detect fresh network attacks as shown in figure 2. The main concern is to make sure that in case of an intrusion attempt, the system is able to detect and report intrusion. IDSs are usually deployed alongside other preventive security mechanisms and there are several reasons that make intrusion detection a necessary part of the entire defense system. First, many traditional systems and applications were developed without security in mind. In other cases, systems and applications were developed to work in a different environment and may become vulnerable when deployed. Intrusion detection uses these protective mechanisms to improve system security. Furthermore, even if the preventive security mechanisms can protect information systems successfully, it is desirable to know what intrusions have happened or are happening, so that we can understand the security threats and risks and thus be better prepared for future attacks (Feng et al., 2014).

The advantages of this service are the “round-the-clock” aspect, in that the system is protected even when the user is asleep or else away from any computer hooked up to the network. Easier to deploy as it does not affect existing systems or infrastructure, Its sensors

can detect many attacks by checking the packet headers for any malicious attack like TCP SYN attack and fragmented packet attack. It monitors traffic in real time. Therefore, network-based IDS can detect malicious activity as they occur. Additionally, the IDS sensor deployed outside the firewall can detect malicious attacks on resources behind the firewall (Kaur et al., 2014).

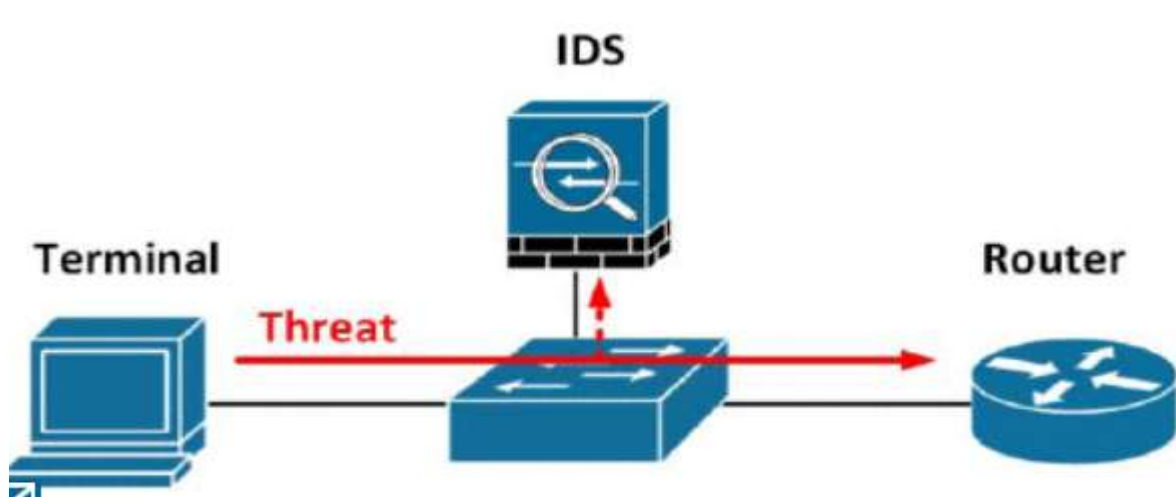


Figure 2: Intrusion Detection System (Hock, & Kortis, 2015)

According to Syujak (2012), the security of the computer network is a very important part to maintain the validity and integrity of the data. Besides, it also guarantees the availability of the services for its users. An attack on the computer network server can occur at any time, either when administrators are working or not working. Thus, the server security system is needed to detect whether each incoming packet is the actual data packets. If the packet is an attacker's packet data, the system is expected to block the attacker's IP. According to Khamphakdee et al., (2014), the network security system and Internet Service Provider (ISP) is an important factor to guarantee the stability, integrity, and data validity. The implementation of the Intrusion Detection System based on Snort can save money when buying the software. This is because snort is an open source software. Testing in the IDS system is done by some attack patterns. It is done to test the ability of snort to detect an attack against the security system. Based on the testing result among the IDS Snort system with port

scan, virus test, buffer overflow, SQL injection, and accessing database, snort can give an alert if there is an attack on the network system.

When a possible intrusion or suspicious pattern is discovered, an alarm is raised by IDS. The system is so-called compromised when the overall network structural design does not relate to the security mechanism. Any attempt at file modification, malicious activity, and unauthorized entrance can be monitored by the system. An IDS operates by monitoring system activity through examining the weaknesses of the system, the integrity of files and analyzing patterns on the basis of past attacks. There is also automatic monitoring of the Internet to seek the most recent threats, which could lead to a future attack. Generally, IDS can be grouped into two categories: signature based and anomaly detection. In a signature-based system, patterns of attack or intruder behaviors are modeled and a system alert sent out once there is a match detected (Aslahi-Shahri et al., 2016).

This system, however, has limitations and is able to detect only known attacks, and as such, there should be a frequent update for the attack signatures. On the contrary, the anomaly detection system considers the behavioral norm of the network or system and creates a baseline profile of normal activities. Then, any activity not matching the system's behavioral norm will be deemed as an intrusion. Anomaly detection systems are capable of detecting the attacks that have been previously known, and therefore, the efficiency of these systems is more than the efficiency of signature-based systems (Ahmed et al., 2016).

2.2.4.2 Snort-IDS

Snort is an open source network intrusion detection and prevention system. It is capable of performing real-time traffic analysis and packet logging on IP networks. It can perform protocol analysis, content searching/matching, and can be used to detect a variety of

attacks and probes, such as buffer overflows, and stealth port scans. Snort is a packet sniffer that monitors network traffic in real-time, examining each packet closely to detect a dangerous payload or suspicious anomalies. Snort is a popular Intrusion Detection and Protection System (IDS/IPS) which use for protecting the system's risk from the attacker. It is an open source lightweight software, was developed by Martin Roesch with C language in 1998 Snort can be installed on almost computer architecture and operating system platform. Furthermore, Snort-IDS also generate an alert in real-time (Naik, 2015).

Snort searches and matches the network traffic's data packet with the rules for checking abnormal data packet traffic. The rules of Snort-IDS are in the form of a single line. It is easy to read and understand and can be modified. Snort-IDS's basic components consist of the Packet Decoder, Preprocessor, Detection Engine, Logging and Alerting System, and Output Modules.

The Snort-IDS utilize the rules matching with the data packet traffic network. Figure 3 shows the basic structure of the Snort-IDS rules which are divided into two logical parts the rule header and the rule option. It contains the criteria definition for matching between a rule and the data packet traffic network. In addition, the action field of the rule header also able to define the type of action such as passes, log alerts. The rule options follow the rule header and they are within a pair of parentheses (Dietrich, 2017). According to Khamphakdee et al., (2014) each option presented the tool that helps the network administrator to make the Snort-IDS rules and alert via Graphical User Interfaces (GUI).

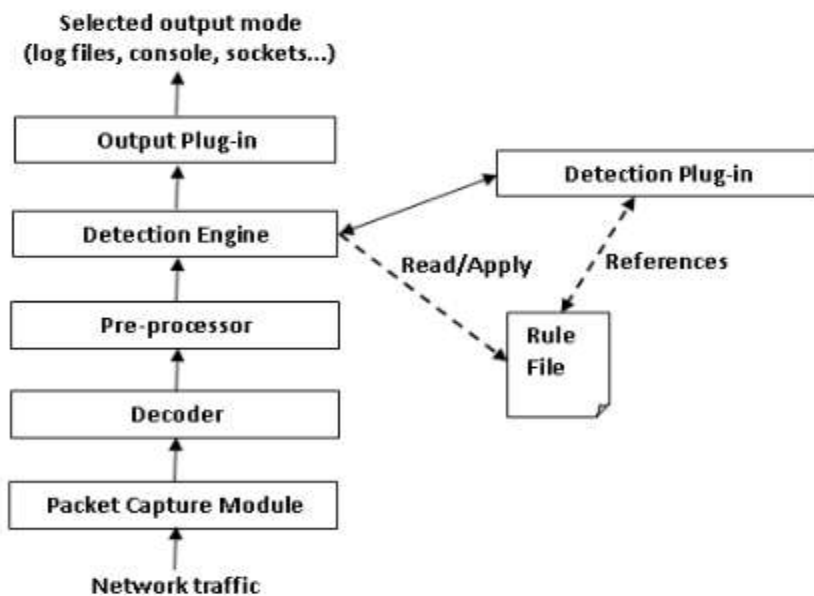


Figure 3: Snort IDS (Khamphakdee et al., 2014)

Snort-IDS are the attacking detection tool, which the researchers around the world interested in. Beside Patel and Sonker (2016) introduced the signature-based development with Snort for analyzing the abnormal connection and they utilize Basic Analysis and Security Engine (BASE) for displaying the generated alert results of the Snort-IDS. Nevertheless, these researches did not improve the rules to increase the efficiency of attacking detection.

According to Veerman and Oprea(2012), different SQL Injection attacks and their solutions based on the SNORT tool was proposed. They captured some SQL Injection attack patterns however, their attack patterns are only extracted signature-based features, which cannot offer protection against more recent attacks. Their approach can detect insider attacks up to 70% of all database attacks such as resource exhaustion, password attack, and malware attacks such as viruses, worms, and Trojan horses.

Warneck (2007) uses many ways for defeating the SQL injection attack to prevent the vulnerabilities related to web attacks by using the SNORT tool for detecting various types of SQL injection attacks in both the database level and the web application level. Dabbour et al.

(2013) presented three types of attacks, such as SQL injection attacks, XSS (Cross-Site Scripting) attacks, and command execution attacks. They also use SNORT IDS for detection and Damn Vulnerable Web Application (DVWA) for evaluating and testing the SNORT rules. AINabulsi et al. (2014), investigated how to alert and detect an SQL injection, XSS, query command injection, and OS command injection attacks on web applications using SNORT tools. They use Samurai-WTF (Web Testing Framework) distribution, Damn Vulnerable Web Application (DVWA), and Security Onion instances for SNORT, in their experimental process. According to Alnabulsi et al., (2014) SNORT tools for detecting SQL injection and XSS attacks were presented. They implemented web attack detection using IDS, which is an algorithm that uses a greedy approach by selecting the best attribute to split the dataset on iteration.

The security system using Snort has some disadvantages, such as detection of flooding data based on size only; the result in the system will not give any action. It can only be used to do Intrusion Detection behind the firewall, connecting it to the Internet generates too many positives. Snort must be twisted to reduce the 'false positives' specific to your environment. Unidirectional Ethernet cables should be used for sensors installation to avoid security concerns (Bul'ajoul et al., 2015).

2.2.4.3 Intrusion Detection Techniques by IDS

The most common intrusion detection techniques used by IDS are based on the behavior of users and signatures of known attacks. To improve the performance of IDS, it is better to use a combination of these techniques.

2.2.4.4 Signature-Based Detection

Signature-based detection is performed by comparing the known information with the database of signatures. A signature can be an earlier defined set of rules or patterns that are

related to known attacks. The signature-based technique is also recognized as a misuse detection technique (Liao et al., 2013). Nevertheless, it is incapable to identify unknown attacks in the cloud (Modi et al., 2013).

2.2.4.5 Anomaly-Based Detection Technique

Anomaly-based detection technique compares current user activities against previously loaded logs of users. It produces a large number of false alarms because of irregular network and user behavior. It also requires huge data sets to train the system for normal user profiles (Kene, & Theng, 2015). An anomaly detection system, unknown attacks can be detected at different levels. Monitoring intrusions from large data becomes difficult at different levels (system, network) of the cloud.

2.2.4.6 Hybrid Detection Technique

The hybrid intrusion-detection-system is the combination of signature and anomaly-based detection method, which is called a hybrid detection technique. The idea behind the implementation of hybrid detection is to detect both management properties with less human interaction known and unknown attacks based on signature and anomaly detection techniques (Patel et al., 2013). Hybrid based IDSs detect intrusions by analyzing application logs, system calls, filesystem modifications password files, binaries, access control lists, and capability databases and other host states and activities (Wang, & Jones, 2017).

2.2.5 Intrusion Prevention System

The Intrusion Prevention System (IPS) is a tradition effort to monitor and secure a cloud-computing system. IPS can respond to detected threats by triggering a variety of prevention actions to tackle malicious activities. IPS is able to prevent a detected threat immediately. Incriminated packets are discarded however a little delay cannot be avoided

because every packet is examined with the IPS subsequently deciding if the packet will be dropped or allowed entry into the secured network (Hock, & Kortis, 2015).

The main functions of intrusion prevention systems are to identify malicious activity and attempt to block or stop the malicious activity. Intrusion prevention systems monitor network traffic or system activities for malicious activity. Intrusion prevention systems are placed in-line and are able to actively prevent and/or block intrusions that are detected (McDougal et al., 2014).

There are four types of prevention systems they are Network (NIPS), NBA, WIPS and Host (HIPS) as shown in Table 1. These systems watch network traffic and automatically take action to protect networks and systems. The drawback of IPS is false positive and negatives. False positive is defined to be an event, which produces an alarm in IDS where there is no attack. False negative is defined to be an event, which does not,produces an alarm when there is an attack takes place. An inline operation can create bottlenecks such as a single point of failure, signature updates, and encrypted traffic. The actions occurring in a system or network is measured by IDS (Vijayarani & Sylviaa, 2015).

It is important to note that after the implementation of mistaken rules the IPS will also discard secure traffic while IDS just creates many mindless reports as shown in Figure 4. Defense systems are required to successfully determine the need for their use before they can be executed, otherwise, the user experience will deteriorate (Carlin et al., 2015). Intrusion Detection systems have become a necessary addition to the security infrastructure of most organizations, precisely because they can stop attackers while they are gathering information about your network (Latha, 2016).

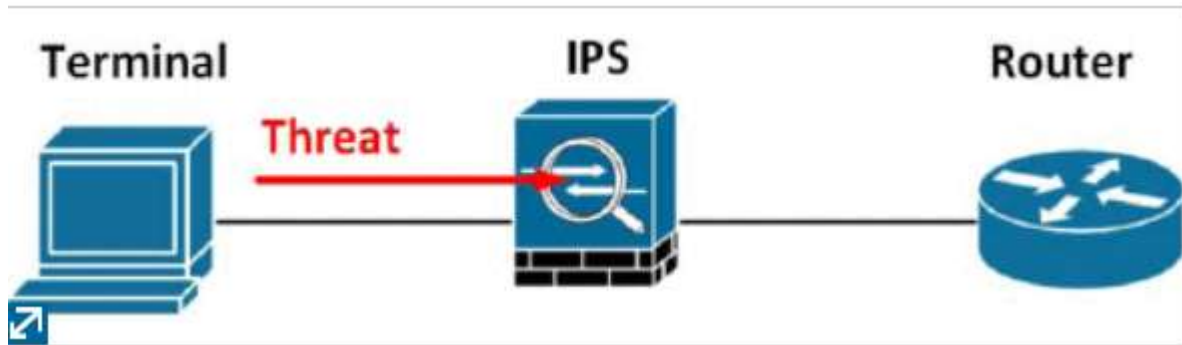


Figure 4: Intrusion Prevention System (Hock, & Kortis, 2015)

IDS can prove to be an invaluable tool in the early detection of malicious activity helping to prevent attacks from succeeding. They can also gather forensic evidence. Nonetheless, traditional IDS are largely ineffective when applied to cloud computing given its openness. Patel et al., (2013) explored the requirements of IDS in the cloud architecture given the ineffectiveness of traditional methods by asking what criteria and requirements should an IDS meet to be deployed on the cloud.

Table 1: Types of Intrusion Prevention System (Scarfone, & Mell, 2012)

Types of IPS	How it is used
Network-based IPS (NIPS)	Monitors traffic in the network and blocks suspicious data stream.
Wireless Intrusion Prevention Systems (WIPS)	Monitor actions in wireless networks. Commonly, it detects wrong configured wireless access points, man-in-the-middle attacks, Mac addresses spoofing
Network Behavior Analysis (NBA)	Analyzes network traffic and look for untypical streams, such as DoS and DDoS attacks.
Host-based Intrusion Prevention (HIPS)	Is resident program, it detects suspicious actions on the computer

2.2.6 Firewall

A firewall is a combination of hardware and software that isolates an organization's internal network from other networks, allowing some packets to pass and blocking others. It functions to avoid unauthorized or illegal sessions established to the devices in the network areas it protects. Firewalls are configured to protect against unauthenticated interactive logins from the outside world. The firewall can be thought of as a pair of mechanisms: one, which exists to block traffic, and the other, which exists to permit traffic as shown in Figure 5. Numbers of firewalls can be deployed in the proper positions of the managed network for cooperative, integrated, and in-depth network security protection (Kaur et al., 2014).

Advantages of Firewalls include preventing traffic, which is not legitimate. Firewalls can filter those protocols and services that can be easily exploited. It helps in protecting the internal network by hiding the names of internal systems from the outside hosts. Firewalls concentrate on extended logging of network traffic on one system (Strohmeier et al., 2014).

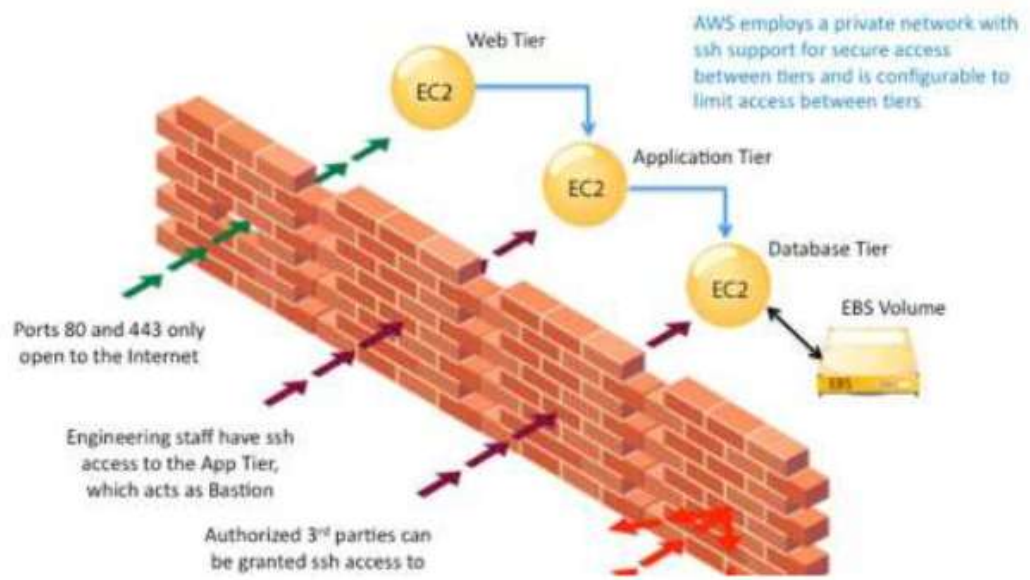


Figure 5: Firewall (Fernandez et al., 2014)

2.2.7 Types of Firewall

There are many different types of firewalls, each of which works in different ways to protect different types of resources, both within data centers and corporate perimeters and outside in the cloud. Here are the most important types of firewalls:

2.2.7.1 Packet - Filtering Router

A packet-filtering router applies a set of rules to each incoming and outgoing IP packet and then forwards or discards the packet. The router is configured to filter packets going in both directions. Filtering rules are based on information contained in the network packet, which includes the source IP address, destination IP address, source and destination transport level address, IP protocol field and interface. The packet filter is typically set up as a list of rules based on matches to fields in the IP or TCP header. If there is a match to one of the rules, then the rule is invoked to determine whether to forward or discard the packet. If there is no match to any rule, then a default action is taken. The default action can either be to discard or forward the packet (Gamage et al., 2016).

2.2.7.2 Application Level Gateways

An application level gateway acts as a relay of application-level traffic. It is also known as a proxy server. The user contacts the gateway using a TCP/IP application, such as Telnet or FTP, and the gateway asks the user for the name of the remote host to be accessed. When the user responds and provides a valid user ID and authentication information, the gateway contacts the application on the remote host and relays the TCP segments containing the application data between the two endpoints. If the gateway does not implement the proxy code for a specific application, the service is not supported and cannot be forwarded across the firewall. Application-level gateways tend to be more secure than packet filters. It is easy

to log and audit all incoming traffic at the application level. The main disadvantage of this type of gateway is the additional processing overhead on each connection (Kaur et al., 2014).

2.2.7.3 Circuit Level Gateways

The circuit-level gateway can be a standalone system or it can be a specialized function performed by an application level gateway for certain applications. A circuit level gateway does not permit an end-to-end TCP connection; instead, the gateway sets up two TCP connections. One connection is set up between itself and a TCP user on an inner host and one between itself and a TCP user on an outside host. Once the two connections are established, the gateway typically relays TCP segments from one connection to the other without examining the contents. The security function consists of determining which connections will be allowed (Marsico, 2015). An advantage of Circuit level gateways is that it is comparatively inexpensive and provides anonymity to the private network. The Disadvantage of Circuit level Gateways is that it does not filter Individual Packets. After Establishing a Connection, an Attacker may take advantage of this (Naik, & Jenkins, 2016).

2.2.8 Flow Chart of a Firewall

The control logic of the firewall system receives incoming and outgoing connections from the network and client machine respectively. In response to a connection request initiating a connection between respective endpoints in the network and client machine. Control logic performs a security assessment comprising obtaining from at least one of the network and client machine information indicative of the security state of the endpoint therein and allows or inhibits the connection in dependence on the result of the security assessment. The security assessment may be performed in accordance with a security policy of the system, and different security assessments may be performed for different connection

requests in accordance with the security policy as shown in Figure 6 (Jansen, & Tanner, 2014).

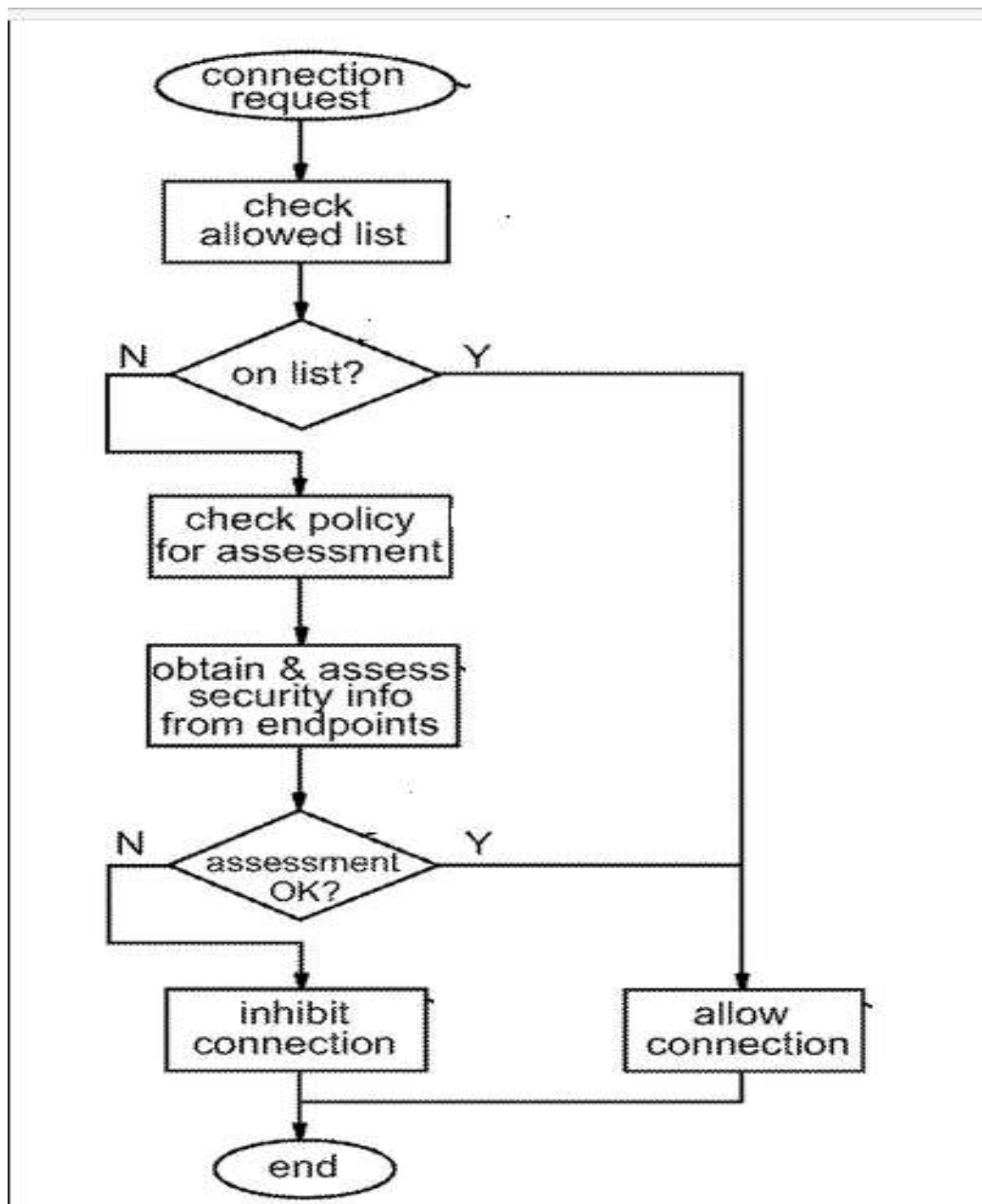


Figure 6: Flowchart of a firewall (Jansen, & Tanner, 2014)

2.3 Weaknesses of Existing Detection Tools

Weakness exploits would end in security breaches or violations of the model security policy causing data leakage or trust issues. Fevre et al. (2012) aver that network

administrators often give staff policies the benefits of doubt because employees do not always break the rules for malicious or vindictive reasons. Rather personnel may not even know that certain actions break the company's policy. Nevertheless, thousands of breaches occur daily and they cost companies millions of dollars. Breaches occur when employees store organization information in third-party cloud services or when they use a blacklisted app, jailbroken phone or other devices that do not meet the organization guidelines.

Bardach and Patashnik (2015) posit that guidelines for devices and software minimize the way people really work and they do not need to go round restrictions. Educating staff on the risks of exposing company data increases employees' satisfaction, staff morale, and convenience. In order to fight IT infrastructure policy violations and gather relevant incidence information in a LAN environment, law enforcement agencies have also started incorporating the collection and analysis of digital evidence data into their infrastructures. Nevertheless, they are yet to consider application deployment and provisioning strategies (Dagada, 2014).

2.3.1 Weaknesses of Intrusion Detection System

Some of the weaknesses of IDS are as follows not an alternative to strong user identification and authentication mechanism. It is not a solution to all security concerns. It will not prevent incidents by itself; it merely helps to uncover attacks. IP packets can still be faked and Human intervention is required to investigate the attack once it is detected and reported. Additionally, False positives occur when IDS incorrectly identifies normal activity as being malicious. False negatives occur when IDS fails to detect malicious activity. It is also susceptible to protocol-based attacks. This means that encrypted packets are not processed by IDS (Chowdhary et al., 2014).

2.3.2 Weaknesses of the Intrusion Prevention System

IDS face difficulties when transferred from traditional networks to cloud-based designs. Issues such as those with their deployment locations and the separation of legitimate traffic from malicious traffic pose challenges with their implementation. IPS does not scale to deal with cloud requirements and does not satisfy the requirements of high-speed networks. IPS requires flow-based, decision making. It is difficult to identify and recognize the analysis of packets in real time traffic. IPS generates a high false alarm rate or positive alarms. There is no uniform standard or metric for evaluating IDS, which can often lead to misleading information as to their effectiveness. Inability to correlate incoming alerts, and it is very difficult to identify internal intrusion attacks given that correctly configuring the systems and implementing organizational policies is a difficult task (Vijayarahi, & Sylviaa, 2015).

2.3.3 Weaknesses of Firewall

According to Hock and Kortis (2015), challenges of the firewall are as follows; use of a set of rules that are manually configured. It cannot react to a network attack nor can it initiate effective counter-measures. Most firewalls do not analyze the contents of the data packets that make up network traffic. It cannot prevent attacks coming from Intranet, this means that it cannot fend off internal attacks. Filtering rules of the firewall cannot prevent attack coming from the application layer. Firewall produces many false positive alarms, it has limited prevention and firewall is created to prevent intrusion from traffics that only pass through it. A firewall makes communication insecure. If an organization allows communication from outside the network it has no ability to be able to scan and prevent the virus.

According to Jansen and Tanner (2014) aside from schemes like the above, the main measures employed today to counter malware attacks are the local installation of scanning

tools like antivirus and spyware tools and local installation of personal desktop firewalls. These have a number of drawbacks. For example, the mechanisms are static: firewalls, for example, rely on decisions of users to create static rules allowing certain executable to access certain network ports and destinations. They also demand user expertise for previous connection requests from local software, the user of the machine is prompted to deny or allow the access request. Very often the user has insufficient expertise to make these decisions, leading to ‘holes’ in the firewall through inappropriate choices of the user. There is the additional difficulty of administration and maintenance: how to place the software on every desktop and how to keep it up-to-date with signatures for example. Further, an operation is mostly based on signatures for known-bad software, so systems are not ready for day zero exploits for which signatures are not yet available. In addition, the placement of these security mechanisms on the local machine allows malware, after a successful compromise, to disable the security mechanisms or hide from them.

Bensefia and Ghoualmi (2011) introduced a contribution consists of designing and developing an intelligent system in order to help the security administrator to exploit, manage and analyze the firewall log files content. The firewall log files trace all incoming and outgoing events in a network. Their content can include details about network penetration attempts and attacks. Eronen and Zitting (2001) presented a tool that helps administrators in analyzing firewall rules. Their presented tool was based on constraint logic programming (CLP) that allows the user to write higher-level operations for detecting common configuration mistakes. One of the weaknesses of packet filtering is that it pretty much trusts that the packets themselves are telling the truth when they say who they are from and where they are going. The firewall also has a weakness of packet filtering, it examines each packet in isolation without considering what packets have gone through the firewall before and what packets may follow. Although network traffic contains huge number of events and a lot of

useful information, most of the current packets filtering techniques exploit the characteristics of filtering rules but they do not consider the traffic behavior in their optimization schemes (Nurika et al., 2012).

The firewall assigns an address to the malicious device and makes the device think it was on a valid host, but would not transfer the communication further up the chain to the host. Stateless Firewall checks all incoming packets individually with the ruleset (Shah, 2015). It does not track the connection nor keeps the state of the traversed packets, rather simply check each packet with Firewall rules and identifies whether the packet is allowed to pass by or not. It assumes that the information within a packet is trustworthy. Hence, one possible breach could be: a TCP SYN-ACK packet can be passed by the Firewall before the Firewall has seen a TCP SYN packet.

2.4 Demonstrate the weaknesses of existing detection tools on policy violation in the cloud

According to Lo et al., (2010) proposed and implemented IDS that worked in a supportive way to oppose the DoS and DDoS attacks. It consisted of four components. The first component performed intrusion detection by collecting and analyzing the network packets. The second component immediately drops the packets and checks whether it is correspondence with the block table rules or not, if packets having no autonomic element manager, autonomic coordinator, and correspondence to these rules are forwarded to the alert clustering module which generates alert for the suspicious packet. The third component blocks the suspicious packets and sends alerts to other IDSs. The fourth component collects alerts and makes a decision about the packet. They could protect the system from a single point of failure attack by deploying the above-proposed IDS. Though, it cannot detect unknown attacks since it uses signature-based detection techniques to detect intrusions.

Vieira et al. (2010), have tested the deployment of IDS at different positions in the cloud to detect DoS attacks. The authors have well-thought-out two scenarios to calculate the IDS performance based on its position in the cloud. Calculation results have shown that detection of DoS attacks using a single IDS instance located close to the Cloud Controller which will significantly increase the load on Cloud Controller. On the other hand, deployment of split instances of IDS at each virtual machine affects only the CPU load of the attacked VM and there is no significant impact on other VMs. The proposed technique is signature based so unable to detect unknown attacks.

According to Golshan and Binder (2016), demonstration detection examination of hard-drive and memory has been a widely used technique to detect the host of malware instances. They proposed a new demonstration detection approach based on a portable executable (PE) format file relationship. This approach has been implemented and validated in HADOOP platform. This approach provided a higher detection rate as well as lower false positive rate. The main drawback of this approach is that its success is based on three assumptions. Most legitimate programs and malware files are in PE format and lie within a windows platform. The number of legitimate files is greater than that of malware files in the user's computer and creating PE format files occasionally happens in a user's computer. Nevertheless, an attacker could exploit any vulnerability in the cloud to attack without following any of these pre-requisites. The authors failed to discuss the consequences of the absence of these pre-requisites such as how efficient this approach would be if one or more of the assumptions are not fulfilled and how much damage the attacker could cause to system or data in absence of these assumptions.

Another model to counter the attack is called 'CloudAV' is provided by Zhang et al., (2014). They gave two main features that make it more efficient, accurate and fast as a malware detection system. One of the features is antivirus as a network service it is a

detection capability by host-based antivirus. It is more efficient and effective as a cloud-network-service. Each host runs a lightweight Process to detect new files and then sends them to network service for quarantine and for further analysis rather than running complex analysis software on each end-host. The second feature is the N-version protection. It is malicious software identification determined by multiple heterogenous detection engines similar to the idea of N-version programming. The notion of N-version protection has been provided in this solution so that the malware detection system should leverage the detection capabilities of multiple heterogeneous detection engines to determine malicious and unwanted files more effectively (Xu et al., 2016).

Nevertheless, the number of false positives encountered during normal operations increase compared to 1-version engines. To manage the false positives, the administrator has to set a trade-off between coverage a single detector is enough to mark a file as malicious and false positives a consensus of a number of detectors is required to mark a file as malicious. The authors concluded that the efficiency of CloudAV through validation in a cloud environment is possible. CloudAV also provides better detection of malicious software and enhanced forensics capabilities.

According to Hatem and El-Khouly (2014) new threat detection through retrospective detection, the approach can improve deployment ability and management. The validation experiment proves that CloudAV provides 35% better detection coverage against threats compared to single antivirus engines and 98% detection coverage of an entire data set of a cloud. Nonetheless, cloud-based security solutions generally suffer from three problems namely security coverage, scalability, and privacy. As malware can be embedded in a large number of file types, attackers may be able to bypass cloud solutions as they are limited to few file types and hence degrade the detection coverage. Additionally, exporting all binaries

or PDF files to the cloud for investigation does not scale and may create a single point of failure by flooding the cloud with benign binaries.

According to Mazzariello et al. (2010), IDS works at the middleware layer and it can detect specific intrusions by using a combination of knowledge and behavior based techniques. They have projected IDS for Grid and Cloud Computing (GCCIDS). In this system, each node identifies the intrusion and generates an alert to another node present in the system since the system works in a cooperative manner. The authors have proposed the behavior-based system by measuring false positives and false negatives and concluded that false negatives are always more than false positives when a similar quantity of data is used as input. On the other hand, they have evaluated the knowledge-based system by using audit data from the system log and communication system and concluded that it is possible to analyze the traffic in real-time if an inadequate number of rules are used for comparison. The authors have not specified implementation details.

Modi et al. (2012) have proposed and implemented a Network intrusion detection system (NIDS) which uses Snort to detect known attacks and Bayesian classifier to detect unknown attacks. NIDS deployed in all servers work in a collaborative approach by generating alerts into the knowledge base and thus making detection of unknown attacks easier. In the given technique, signature-based detection is followed by anomaly-based detection, since it detects just unknown attacks. Conversely, the detection rate is increased by sending alert to other NIDS deployed in the cloud.

According to Riaz et al. (2017) grid and Cloud Computing Intrusion Detection System (GCCIDS), this is designed to cover the attacks that network and host-based systems cannot detect. Their proposed method uses the integration of knowledge and behavior analysis to detect specific intrusions. Conversely, the proposed prototype could not discover

new types of attacks or create an attack database, which must be considered during implementing IDS.

According to Hu et al. (2014), enforcing a conflict-free security policy in cloud environments based on SDN has been studied in Flowguard. These works deal effectively with direct conflicts by rejecting the policy and implementing role-based and signature-based enforcement to ensure applications do not circumvent existing security policy. However, a flow can be defined in multiple layers, where traditional policy checking approaches do not consider; for example, indirect security violations, partial violations or cross-layer conflicts cannot be handled. Additionally, they appear not to fully leverage the SDN paradigm that lets flow rules do traffic shaping in addition to implementing accept/deny security policy.

Chang et al. (2013a) and Chang (2015) proposed their Cloud Computing Business Model (CCBF), which has four major components and compiles a summary of successful deliveries and case studies of Cloud Computing. However, there is no detailed information from the design to implementation of service delivery. Due to this reason, the next phase of work known as Cloud Computing Adoption Framework (CCAF) was developed (Chang et al., 2013b; Ramachandran, & Chang, 2014). CCAF emphasizes more on the practical implementation, service delivery and resolution of problems rather than presenting the conceptual framework. Nevertheless, there is a lack of demonstrations on security, which is an important aspect of Cloud Computing service to ensure all services are well protected. Huang et al. (2013) presented a low reflection ratio mitigation system that consists of source checking, counting, attack detection, turning test and question generation modules. This system is designed to be implemented before the IaaS. This system considers the computational efficiency and overheads in the implementation and their effect on legitimate users. A blacklist, white list, block list and unknown are used to categories incoming packets based on IP addresses; administrators using APIs maintain these. Such APIs open the system

to malicious manipulation from insiders. The system shows an operational degradation of 8.5% when monitoring traffic against a blacklist of 100,000 addresses.

The system proposed by Fujinoki (2013) addresses some of the limitations of overlay networks in hiding the location of target servers with gateway routers. It protects clouds from insider attacks and compromised user host machines. There is no need for migration of network items to other hosts. The approach removes the need to monitor all network traffic resulting in lower computational overhead. Each proxy node contains a Bloom filter, which is a data structure that can efficiently test for the presence of certain values. When an attack warning is issued, more user proxy machines are deployed with the number of users assigned to each being halved until attackers are identified. Published simulation results are very promising; nevertheless, real-world testing is needed to achieve realistic performance measurements.

According to Bulajoul et al. (2015), a real network to present experiments that use a Snort NIDPS was designed. Their experiments demonstrated the weaknesses of NIDPSes, such as the inability to process multiple packets and propensity to drop packets in heavy traffic and high-speed networks without analyzing them. They tested Snort's analysis performance, gauging the number of packets sent, analyzed, dropped, filtered, injected, and outstanding.

Jaiganesh et al. (2013) recommended a novel backpropagation model for intrusion detection. This method makes training pair with a combination of input and equivalent targets were generated and implemented into the network. Performance success can be measured by a false alarm and detection rate. The detection rate was proven to be less than 80% for U2R, R2L, DoS and Probe attacks. Nevertheless, the major issue of the method was found to be inefficient to detect hidden attackers present in the system. Devikrishna and Ramakrishna (2013) used MLP (Multi-Layer Perception) architecture for intrusion detection that detects

and classifies attacks into six types. MLP method was considered as a failure model due to irrelevant output. In the present paper, we have tried to overcome this query and to establish a better detection technique.

2.5 Develop a model to detect and identify policy violation in the real-time traffic

According to Gul and Hussain (2011), an efficient model that uses multithreading technique for improving IDS performance within the Cloud computing environment to handle a large number of data packet flows was suggested. The proposed multi-threaded NIDS is based on three modules namely capture module, analysis module and reporting module. The first one is responsible for capturing data packets and sending them to analysis part which analyzes them efficiently through matching against a pre-defined set of rules and distinguishes the bad packets to generate alerts

Finally, the reporting module can read alerts and immediately prepare an alert report. Consequently, the proposed system can outperform the hybrid system of in terms of preventing the attack from conducting any bad action through blocking the event and saving that threat with the other signatures in order to be observed by Signature-Based Intrusion Detection for next time so that it can be detected earlier (Alsafi et al., 2012).

In SDN IPS Xing et al. (2014) presented a Software Defined Network (SDN) based Intrusion Prevention System solution. It is a full lifecycle solution including detection and prevention in the cloud. A new IDPS architecture is proposed based on Snort-based IDS and Open vSwitch (OVS). The authors have compared the SDN based IPS solution with the traditional IPS approach from both mechanism analysis and evaluation.

According to Ferreira et al. (2017), a solution of monitoring data at runtime and feeding it back into the service registry to adjust descriptions and make contract template derivation as a more realistic process needs to be implemented. Emeakaroha et al., (2012)

introduced the detection SLA violation infrastructure (DeSVi), the narrative architecture for monitoring and detecting SLA violations in cloud computing infrastructures. The main components of the architecture are the automatic VM deployer, responsible for the allocation of resources and for mapping of tasks, application deployer, responsible for the execution of user applications, and LoM2HiS framework, which monitors the execution of the applications and translates low-level metrics into high-level SLAs.

According to Hu et al. (2014), a comprehensive framework FLOW GUARD to accommodate design requirements was proposed. FLOWGUARD addresses several significant limitations in building SDN firewalls to facilitate accurate detection as well as the flexible resolution of firewall policy violations in dynamic OpenFlow networks along with a variety of toolkits for visualization, optimization, migration, and integration of SDN firewalls as shown in figure 7. Flow packet violations can be handled by using the traditional technique for firewall packet filtering. Conversely, it is challenging to deal with a flow policy violation, since both firewall and flow policies support wildcard rules. Moreover, in an OpenFlow network, the header fields of flow packets could be dynamically changed when the packets traverse the network as shown in Figure 8. Thus, to support accurate violation detection and enable network-wide access control, a firewall application needs to not only check violations at the ingress switch of each flow but also track the flow path and then clearly identify both the original source and final destination of each flow in the network.

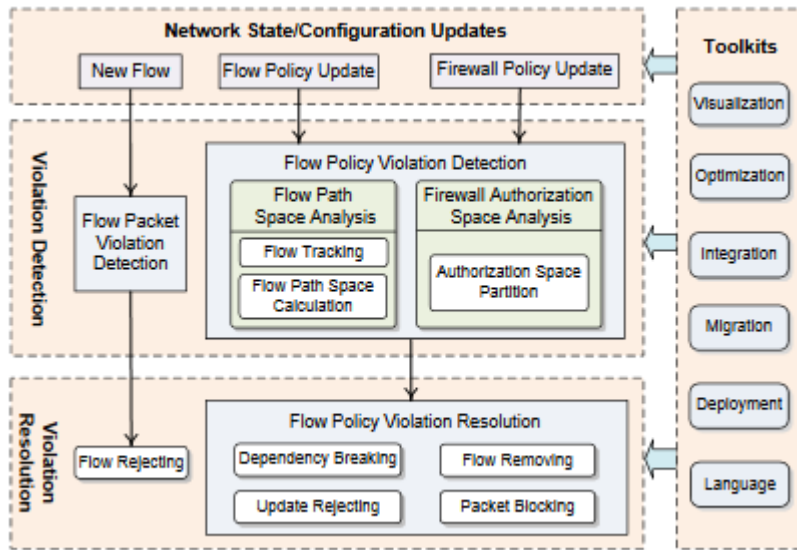


Figure 7: FlowGuard framework (Hu et al., 2014)

According to Bleikertz et al. (2014), an infrastructure cloud consists of (virtualized) computing, networking, and storage resources, which are configured through a management host and its well-defined interface. The system model of this work is poised towards a differential analysis based on change events issued by the management hosts when the infrastructure is re-configured. In figure 7, the analysis system uses these change events to continuously update a graph representation of the infrastructure, the Realization model, which is used for subsequent analysis. As long as the management host issues the events correctly, the model covers malicious adversaries, insiders, and externals alike.

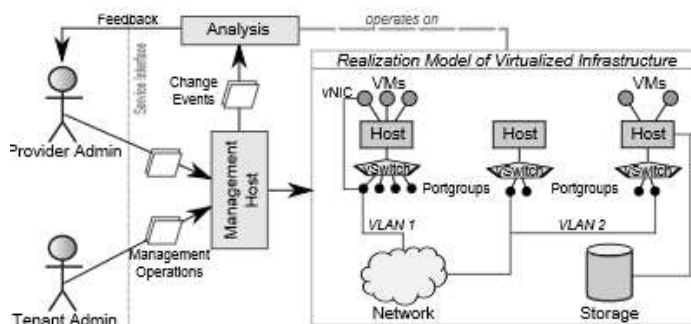


Figure 8: Virtualized Computing (Bleikertz et al., 2014)

According to Jabez and Muthukumar (2015) to develop an IDS based on anomaly detection model that would be precise, not easily cheated by small variations in patterns, low in false alarms, adaptive and be of real time. The proposed system model was the intrusion packets are received from the internet then SNORT is used to collect the datasets. Initially, the features extracted from data packets then forwarded to our proposed IDS. Then, the proposed IDS computes the distance between the extracted features and the trained model. Here, the trained model consists of big datasets with distributed storage environment to improve the performance of Intrusion Detection System. Thus, the outlier value is greater than the specified threshold then it generates the false alarm.

According to Varadharajan and Tupakula (2014) system models involved cloud service providers which included cloud system administrators, tenant administrators (or operators) who manage the tenant virtual machines, and tenant users (or tenant's customers) who use the applications and services running in the tenant virtual machines. Cloud providers are entities such as Amazon EC2 and Microsoft Azure who have a stake in protecting their reputations. The cloud system administrators are individuals from these corporations entrusted with system tasks and maintaining cloud infrastructures, who will have access to privileged domains. They assumed that as cloud providers have a stake in protecting their reputations and resources, the adversaries from the cloud provider perspective are malicious cloud system administrators. In determining the threat model, they needed to look at the different types of attacks that are possible in such a configuration. The circle in Figure 9 showed the source of the attack and the arrowhead shows the target of the attack. They identified three domains in our architecture that are relevant to the threat model. There is the tenant domain comprising tenant administrators and tenant users. Each tenant has its own tenant domain. There is the cloud system domain, which consists of cloud system administrators and the VMM platform with its privileged domain and hardware. Then there is

the cloud cluster domain comprising cloud system domains that constitute the cloud infrastructure.

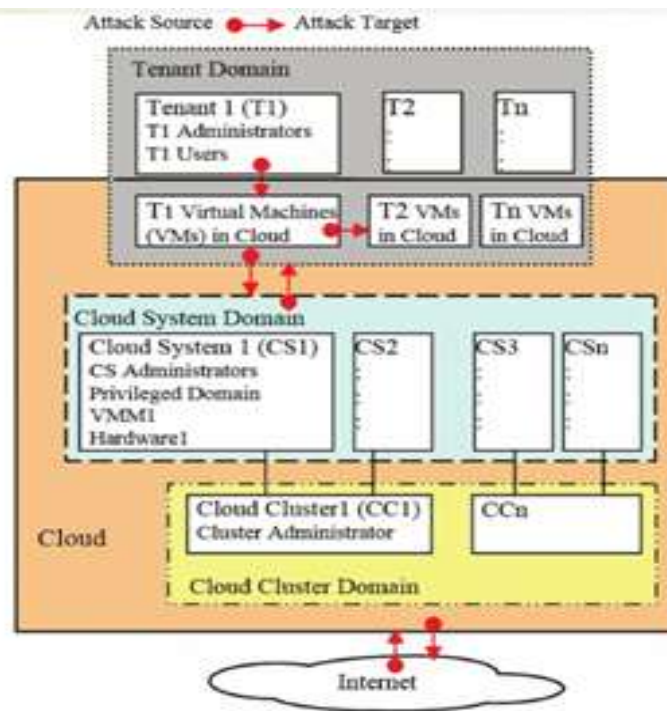


Figure 9: Threat Model (Varadharajan, & Tupakula, 2014)

2.6 Cloud computing crimes

An attack can be anything that can harm a system or cloud network. In terms of specific attacks, which can affect some Cloud services and resources, Patel (2013) listed the possibility of flooding attacks, the user to root attacks, port scanning, backdoor attacks and attacks on the VM or hypervisor. Examples of Cloud resources being used as a tool to commit crimes include the use of the Amazon EC2 instance in 2009 as a command and control server for Zeus botnet (Goodin, 2009). This resulted in the second-largest online data breach in the U.S (Galante et al., 2011). Another example occurred in 2014 when Amazon Web Service (AWS) account was hijacked and extra instances were launched to mine Bitcoins (Rashid, 2014). Examples of the Cloud being the target of crime include the Distributed Denial of Service (DDoS) attack on Bitbucket, a hosting service website (Noehr,

2011), while Sony's Play stations network and Microsoft's Xbox Live services suffered DDoS attacks in 2014 (Paganini, 2014). In addition, Rack space, a Cloud computing service provider suffered a DDoS attack on 21 December 2014, which lasted 12 hours (Martin, 2014). These examples show the susceptibility of the Cloud to crime and that there is, therefore, a need for investigative strategies for a Cloud environment. The attacks can also cause the policy violation by using affected machine to access the network. The most common attacks that affect the cloud are as follows.

2.6.1 Virtual Machine Attacks

Attackers effectively control the virtual machines by compromising the hypervisor. The most common attacks on the virtual layer are Xen – based Host system firewall and its extensions and NOVA a micro hypervisor-based secure virtualization architecture that allows hackers to manage host through the hypervisor. Attackers easily target the virtual machines to access them by exploiting the zero-day vulnerabilities in virtual machines. This may damage several websites based on a virtual server (Ibrahim et al., 2016).

The solution to prevent VM rollback attack is based on disabling the suspend and resume functionalities of the hypervisor. The suspend/resume feature is powerful for virtualization and disabling it will not provide a better solution. Another limitation of this solution is the excessive user interaction with the cloud system. It requires end users to get involved during VM booting, suspending and resuming. This means that the system needs to ask for permission every time it reboots, migrates or suspends a VM, which makes it inconvenient and impractical. In this solution, only the end user can tell whether a rollback is malicious or not by auditing the log of VM activities. Although this solution has minimized the user involvement compared to the Hyperwall, the changing infrastructure of cloud

computing still demands the autonomous working of VM operation with some user involvement (Szefer, & Lee, 2012).

2.6.2 U2R (User to Root Attacks)

The attacker may hack the password to access a genuine user's account, which enables him to obtain information about a system by exploiting vulnerabilities. This attack violates the integrity of cloud-based systems. These attacks are exploitations in which the hacker starts off on the system with a normal user account and attempts to abuse vulnerabilities in the system in order to gain superuser privileges (Aljawarneh et al., 2018). The proposed model for the detection of User to Root attacks. This model consists of three Phases, in each phase, a set of activities have been carried out. In the first phase KDD CUP 99 data set is taken, from which the User-to-Root data set is created. User-to-Root data set contains 52 samples with 41 attributes.

2.6.3 Insider Attacks

The attackers can be authorized users who try to obtain and misuse the rights that are assigned to them or not assigned to them (Pandeewari, & Kumar, 2016). An insider attack is a malicious attack perpetrated on a network or computer system by a person with authorized system access. Insiders that perform attacks have a distinct advantage over external attackers because they have authorized system access and may be familiar with network architecture and system policies/procedures. In addition, there may be less security against insider attacks because many organizations focus on protection from external attacks. An insider attack is also known as an insider threat. Insider attacks can affect all computer security elements and range from stealing sensitive data to injecting Trojan viruses in a system or network. Insiders also may affect system availability by overloading computer network storage or processing capacity leading to system crashes. Internal intrusion detection systems (IDS) protect

organizations against insider attacks, but deploying such systems is not easy. Rules must be established to ensure that employees (Khurana et al., 2014) do not trigger unintended attack warnings.

2.6.4 Denial of Service (DOS) Attack

In cloud computing, the attackers may send a huge number of requests to access virtual machines thus disabling their availability to valid users, which is called DoS attack. For example, one can launch a DoS attack by just using the ping command. This will result in sending the victim an overwhelming number of ping packets. If the attacker has access to greater bandwidth than the victim does, this will easily and quickly overwhelm the victim (Chowdhary et al., 2014). The attack targets the availability of cloud resources.

2.6.5 Port Scanning

Ports are like little doors on your system. Most packets leaving your machine come out of a certain door. They are destined for another door on another system. Transport layer protocols, including the Transmission Control Protocol (TCP), User Datagram Protocol and the Stream Control Transmission Protocol, use ports, which, taken together with an IP address, are used to identify the processes running on a networked host to which a packet is sent. Transport layer protocols in the TCP/IP stack can use any of up to 65,535 different ports to listen for and respond to requests from remote hosts (Gould, & Danforth, 2016).

A port scan attack, therefore, occurs when an attacker sends packets to your machine, which can vary the destination port. The attacker can use this to find out what services you are running and to get a pretty good idea of the operating system you have. Most internet-facing systems get scanned every day, though as long as you harden your firewall and minimize the services allowed through it, these attacks shouldn't worry you (Ahanger, 2014).

2.6.6 Backdoor Path Attacks

According to Gonzales et al. (2017), hackers continuously access infected machines by exploiting passive attacks to compromise the confidentiality of user information. Hackers can use a backdoor path to get control of infected resource launches a DDoS attack. This attack targets the privacy and availability of cloud users. Backdoor Use in Targeted Attacks,” applications that allow for remote access to computers known as backdoors are often used for targeted attacks. In these types of breaches, hackers leverage backdoor programs to access the victim’s network. The benefit of this attack vector is that the backdoor itself can help cyber criminals break into the infrastructure without being discovered. Backdoors not only provide a disguised point of entry for hackers but can also offer a number of strategies for the intrusion. Trend Micro’s report noted that these include:

2.6.7 User Spoofing

Spoofing attacks can occur in one of two ways. First, there is local spoofing a type of attack carried out when the attacker and the victim are on the same subnet. Of the two types of spoofing attacks, we’ll discuss here, this is by far the easier. The attacker has the ability to sniff traffic on the network, and thereby uncover key pieces of information needed to launch the attack. While some of the techniques used to initiate this type of attack occur in the transport layer, it is important to understand that attackers will corrupt the data stream, spoof addresses, and attempt to inject sequence numbers into packets that will help them gain control of the communication session. The second way in which this attack may be launched is by means of blind spoofing. This is a much more sophisticated and advanced attack. When launched in this manner, the attacker is not on the same local subnet. This means many of the pieces of information that the attacker will need to be successful are not available. These key parameters must be guessed. Most modern OS use random sequence numbers making this

type of attack difficult to launch. While still possible to launch, the zenith of these attacks has passed (Krombholz et al., 2015).

An attacker may send a forged ARP packet containing a false IP-to-MAC address binding to a gateway or a host. The forged ARP packet sent from Host A deceives the gateway into adding a false IP-to-MAC address binding of Host B. After that, normal communications between the gateway and Host B are interrupting (Maynard et al., 2014).

2.6.8 Penetration Attack

Penetration attacks contain all attacks, which give the unauthorized attacker the ability to gain access to system resources, privileges, or data. One common way for this to happen is by exploiting a software flaw. This attack would be considered a penetration attack. Being able to arbitrarily execute code as root easily gives an attacker to whatever system resource imaginable. In addition, this could allow the user to launch other types of attack on this system or even attacks from other systems from the compromised system (Singhal, & Ou, 2017).

According to Shah and Mehtre (2015) penetration test, also known as a pen test, is a simulated cyber-attack against your computer system to check for exploitable vulnerabilities. Pen testing can involve the attempted breaching of any number of application systems, application protocol interfaces (APIs), frontend/backend servers to uncover vulnerabilities, such as sanitized inputs that are susceptible to code injection attacks. Insights provided by the penetration test can be used to fine-tune your WAF security policies and patch detected vulnerabilities.

2.6.9 Malware Injection Attacks

Malware injection attack refers to a manipulated copy of the victim's service instance, uploaded by an attacker to cloud so that some service requests to the victim's service are processed within the malicious instance. An attacker can get access to the user's data through the malware injection. The attacker actually exploits its privileged access capabilities in order to attack that service security domain (Papp et al., 2015). The incidents of this attack include credential information leakage, user private-data leakage and unauthorized access to cloud resources. The challenge does not only lie in the failure to detect the malware injection attack but also in the inability to determine the particular node on which the attacker has uploaded the malicious instance (Kirat et al., 2014).

According to Liu and Chen (2010), retrospective detection examination of hard-drive and memory has been a widely used technique to detect the host of malware instances. They proposed a new retrospective detection approach based on a portable executable (PE) format file relationship. This approach has been implemented and validated in HADOOP platform. This approach provides a higher detection rate as well as lower false positive rate. The main drawback of this approach is that its success is based on three assumptions. Most legitimate programs and malware files are in PE format and lie within a windows platform. The number of legitimate files is greater than that of malware files in the user's computer and creating/writing/reading PE format files occasionally happen in a user's computer. However, an attacker could exploit any vulnerability in the cloud to attack without following any of these pre-requisites. The authors failed to discuss the consequences of the absence of these pre-requisites such as how efficient this approach would be if one or more of the assumptions are not fulfilled and how much damage the attacker could cause to system or data in absence of these assumptions (Li, & Gaudiot, 2019).

2.6.10 Cross VM Side-Channel Attacks

VM side-channel attack is an access-driven attack in which an attacker VM alternates execution with the victim VM and leverages the processor caches to infer the behavior of the victim. It requires that the attacker resides on a different VM on the same physical hardware as that of the victim's VM. Zhang et al., (2014) discussed a comprehensive example of how to collect information from a target VM through cross VM side-channel attack. One incident of side-channel attacks is the timing side channel attack which is based on measuring how much time various computations take to perform. Successful modulation of this measured time may lead to leakage of sensitive information about the owner of the computation or even the cloud provider. Timing channels are especially hard to control and pervasive on clouds due to massive parallelism. Moreover, timing side-channel attacks are hard to detect since they do not leave trails or raise any alerts. Cloud customers may not have the authorization to check for possible side channels from other cloud mates obviously due to privacy concerns.

2.6.11 Theft of Service Attacks

The Theft of Service attack utilizes vulnerabilities in the scheduler of some hypervisors. The attack is realized when the hypervisor uses a scheduling mechanism, which fails to detect and account for the Central Processing Unit (CPU) usage by poorly behaved virtual machines. This failure may further allow malicious customers to obtain cloud services at the expense of others. This attack is more relevant in the public clouds where customers are charged by the amount of time their VM is running rather than by the amount of CPU time used. Since the Virtual Machine Manager (hypervisor) schedules and manages virtual machines, vulnerabilities in the hypervisor scheduler may result in inaccurate and unfair scheduling. These vulnerabilities mainly result from the use of periodic sampling or low-

precision clock to measure CPU usage: like a train passenger hiding whenever ticket-checkers come for tickets (Khalil et al., 2014).

In the theft of Service attack, the hacker ensures that its process is never scheduled when a scheduling tick occurs. The common incidents of this attack include using cloud-computing services for a long period while keeping it hidden from the vendor and using cloud computing resources storage system or OS platform for a long period without representing it in a billing cycle. A countermeasure to this attack has been provided by modifying the schedule to prevent the attack without sacrificing efficiency, fairness or I/O responsiveness. These modifications do not affect the basic credit and priority boosting mechanisms. The modified schedules are the exact scheduler; uniform scheduler; priority scheduler and Bernoulli scheduler. The main differences among these schedules are in the scheduling and monitoring policies and in time-interval calculations. The experiment conducted by Shea et al., (2014) found out that modified schedulers provide accurate and fair scheduling. The modifications in hypervisor are shown to be beneficial, as compared to Xen hypervisor currently running in Amazon Elastic Compute Cloud EC2.

Kadayiruppu (2014) states that using a new instance of cloud-to-user surface in the victim machine monitors the scheduling of parallel instances suggested another theoretical countermeasure. Then, the outputs of both the attacker and the legitimate instances are compared. A significant difference in results is reported to the responsible authorities as an attack. This solution has not been validated or verified by authors and does not provide any guarantee for a beneficial result. There are other solutions provided for hypervisor scheduling but they are only limited to improving other aspects of virtualized I/O performance and VM security such as CPU-bound issues. These studies do not examine scheduling fairness and accuracy in the presence of attackers, which is the backbone for the theft-of-Service attack (Dall, & Nieh, 2014).

2.7 Review of Models Developed

Several studies have been conducted previously that aimed to integrate IT functions of Public and private cloud computing. Most papers discussed private cloud computing, public cloud computing, and hybrid. According to Vaquero (2011), there is effectiveness when using services cloud such as IaaS and PaaS in educational fields, especially in teaching advanced Computer Science courses. The Blue Sky cloud framework was presented by Goyal (2014) to implement a cloud that supports scalable and cost-efficient for the E-learning system for basic education in China.

According to Sodhi and Prabhakar (2011), pure autonomous system architecture based on IaaS where control of cluster nodes is fully autonomous was presented. This model uses real-time information from cluster nodes and decentralizes the policy management from the master node to other working nodes. It has several main components namely cloud controller, a gateway for clients into cloud, which determines the suitable node to run VM that satisfies client's needs, Cloud agent an intelligent software component that responds to the queries of the cloud controller regarding the availability of VM configuration for a specific lease duration. It also contains VM foundry, VM image repository interface dedicated to answering queries for particular VM configurations and it creates the one-time-URLs for the VM image. The cloud agent is further based on several components including request handler, VM manager, policy manager, capability manager, and data store.

The key point, which differentiates this system from other IaaS based systems, is that it consists of decentralized policy management rather than based on master-slave relationship architecture that provides a bottleneck issue. If a problem occurs in the master, it may cause the system to function abnormally or even to shut down. This issue has been avoided through the decentralization of policies to other nodes. Workload distribution mechanisms for IaaS are static. Decentralization needs to be autonomous, due to rapid change in cloud

infrastructure. Decentralization of policy management among different nodes will increase reliability and security (Sodhi, & Prabhakar, 2011).

Bursztein et al. (2011), worked out a systematic study of existing visual CAPTCHAs based on distorted characters that are augmented with anti-segmentation techniques. Applying a systematic evaluation methodology to 15 current CAPTCHA schemes from popular web sites, they found that 13 are vulnerable to automated attacks. They tested the efficiency of their tool Decaptcha against real CAPTCHAs. To achieve such a high success rate they developed the first successful attacks on CAPTCHAS that use collapsed characters (eBay and Baidu). Only Google and ReCAPTCHA resisted the attack attempts, and they reached some informative understanding of why they could not break them. Because of DeCAPTCHA genericity, they were able to break 7 of these 15 schemes without writing a new algorithm. The result of the analysis shows that the state-of-the-art anti segmentation techniques, state of the art anti-recognition techniques, and CAPTCHAS used by the most popular websites were evaluated. The limitation with this work is that because some features that are ineffective against automated attacks but counterproductive for humans are used. It makes it difficult to be able to break all the schemes. More so, some anti-segmentation techniques are not used.

Chaware (2011) proposed a secured system for banking applications using honeypots and IDS. The honeypot used in this system is the low interaction and high interaction honeypot. The system is implemented in such a way that the users or attackers will either access the network via the Internet or direct. Within a LAN, IDS with honeypot and a centralized server with database layers are being connected. Once the user gets access to the network, all its interactions low or high will be monitored by the IDS and make a log file for that user. IDS will decide to make a user as blacklisted or not, also the server's data will be checked for integrity and identify the source of the user. Database layers will also be checked

for integrity by the system. The proposed banking system divides the internal database into three layers as a) public database (b) main Database and (c) dummy database. Their work revealed that Honeypots have the ability to catch new hacker toolkits and scripts, and are able to reduce the effectiveness of these tools in the wild by allowing security practitioners the capability to analyze these new tools. The limitation to this system is first, in their implementation, they did not include the latest rules for virus and worm detection. More so, the protocols have not been emulated.

Linora and Barathy (2014) proposed an intrusion detection system that depends on the honey pot. They built the models of normal behavior for multitier web applications considering both front-end requests and backend database queries. It provides a container-based IDS with multiple input streams to produce the alerts and can identify a large number of attacks with a minimal false positive rate. This achieves better characterization of the system for anomaly detection and it is more effective for both the static and dynamic web service. The result of the work shows their approach is feasible and effective in reducing both false positives and false negatives. The only problem associated with the work is that IDS works on the assumption for abnormal behavior. The IDS did not indicate the mechanism it will employ for detecting whether it is malicious activities or not.

Dhopte and Chaudhari (2014) proposed Genetic Algorithm Intrusion Detection System (GAIDS), which consists of two phases; Preprocessing and learning phase. The model is an algorithm that was used for detecting four major attacks Denial of Service DOS, User to the root (U2R), Remote to Local (R2L) and probe data set. The result of the algorithm provides a high rate of the rule set for detecting different types of attacks. Their system is more flexible for usage in different application areas with proper attack taxonomy. The use of Genetic algorithm with IDS gives a good result, but one limitation is that it was

not able to handle the sharp boundary problems. For increasing accuracy for intrusion detection combination of fuzzy data mining with GA will be more powerful.

Ogwen et al. (2014) proposed the design and developed an Intrusion Detection System. They also designed a port scanner to determine potential threats and mitigation techniques to withstand these attacks. Implement the system on a host and Run and test the designed IDS. In the project, they set up to develop a Honey Pot IDS System. It makes it easy to listen on a range of ports and emulate a network protocol to track and identify any individuals trying to connect to your system. The IDS will use the following design approaches: Event correlation, Log analysis, Alerting, and policy enforcement. The result of their work attempted to identify unauthorized use, misuse, and abuse of computer systems. The limitation to their work is that it cannot presently contact (raise an alarm) an individual away from his/her PC and there is a need for user sanitization.

Kondra et al. (2016) proposed a new approach which uses the virtualization technique to overcome the existing security problem, it overcomes the limitation of honeypots from single network detection to network across the organization and improves the existing security design to waste the attackers' time as much as possible to get the best useful information. The objective of the work was to analyze the performance of different honeypots based on intrusion detection systems and get the best possible data about the attack and relevant information. When honeypots were implemented, the log file was generated. With the help of the data gathered, it was found that most of the attacks were on protocols, which are based on TCP/IP. HTTP port was one of the most vulnerable ports. Another vulnerable port found was FTP port. It was also found that the number of vulnerabilities increased when this port was opened. The limitation identified is that real-time detection and prevention system to minimize the attack and sources was not achieved.

Muthurajkumar et al. (2013), introduced an intrusion detection model that used a combination of fuzzy SVM and feature selection algorithms to produce high detection rates and minimal false positives. Vieira et al. (2009) proposed a grid and cloud computing intrusion detection system (GCCIDS) that combats attacks by using both signature-based and anomaly-based techniques to detect intrusions. In order to train the system, the authors employed neural network classification algorithms, and the resulting system boasted low processing overhead and satisfactory performance for real-time implementation. Singh et al. (2014), implemented a forest for peer-to-peer botnet detection that proved adept at classifying malicious traffic on a cluster, with low false positive rates and considerable precision and recall.

Malav et al. (2016), proposed a system, which combined specific features and services of IDS, IPS, and Honeypot. Because various exploits were being used to compromise the network, these exploits are capable of breaking into any secured networks. In order to increase the efficiency of network security, they introduce Honeypot. Honeypot detects attacks with the help of IDS; trap and deflect those packets sent by attackers. The result of their work indicates that the system handles multiple clients using the concept of honeypot. An intrusion detection system (IDS) monitors the whole network and looks for the intrusion. When an intrusion occurs, the honeypot is activated. This activated honeypot diverts the traffic to dummy/virtual servers and backtracks the source (IP address) to the origin of that attack. The drawback of the system is that since it supports multiple clients including an attacker the system can easily be compromised.

Yesugade et al. (2016), proposed a combination of the features, functions, and methodology of IDS, IPS, and Honey pot. This is to make IDS more effective, accurate and responsive. Honeypot, IDS, and IPS are eventually deployed on the gateway for analyzing incoming network traffic. The main server will be connected to an Internet Service Provider (ISP) through the external router. All incoming packets from the external network will be

first directed to the mirror server Honeypot to capture the logs. The result of their work shows that the proposed system is more stable and precise on the operating system platform. The system has introduced a sophisticated and interactive user-friendly interface to configure and monitor the software and also to analyze and log the behavior of the intruder and intruding events. The only drawback of the proposed system in its detection module did not include how the IDS should be capable of detecting intrusion by spyware since it is evident that CAPTCHA on the IPS can be broken by spyware.

Ashwini et al. (2017) proposed the implementation of middle interaction production honeypot. Their main goal is to secure the server-side using honeypot from the attackers. The result of the work indicates that Clients can communicate to the servers through the honeypot only. The client has the fake IP address of the honeypot and not the servers. If the client is a genuine client then its request goes to a honeypot. Honeypot changes its IP address and forwards the request to the original server. After that, the server gives a response to a honeypot. Again, honeypot changes its IP address and sends a response to the client. If the client is a fake client, the attacker will be tracked, located, identified and saves information about attackers at the honeypot. Though it is an attacker it gives a response to make them fool. In all these scenarios, security is maintained. The limitation here is that if no attacker comes in, the honeypot system becomes useless. The need for other established security tools such as IDS and IPS to be integrated with the honeypot becomes imperative.

An architecture based on the hybrid cloud model, which uses both the public and private clouds named as university Ucloud. It consists of two main parts the Cloud Management System and the Hybrid Cloud. The cloud management system included the following components, security management, performance monitoring, scheduler, resource allocator. Sqalli et al., (2012) proposed Ucloud architecture that is simulated using CloudSim. The evaluation of this architecture in terms of improvement on productivity in the university

is performed for two separate scenarios. In the first one, the number of tasks is kept constant; and in the second, the number of tasks is changed. The results obtained are encouraging and support the use of a hybrid cloud solution for a university. Nevertheless, the proposed architecture in this research targets private cloud-only. The public cloud is used only to get a better performance or when the load is too high for the private cloud, therefore not used all the time. The results show that high performance can be obtained while keeping the cost low as shown in Figure 10. However, the challenges of public cloud security, confidentiality, privacy were not addressed on this Ucloud architecture.

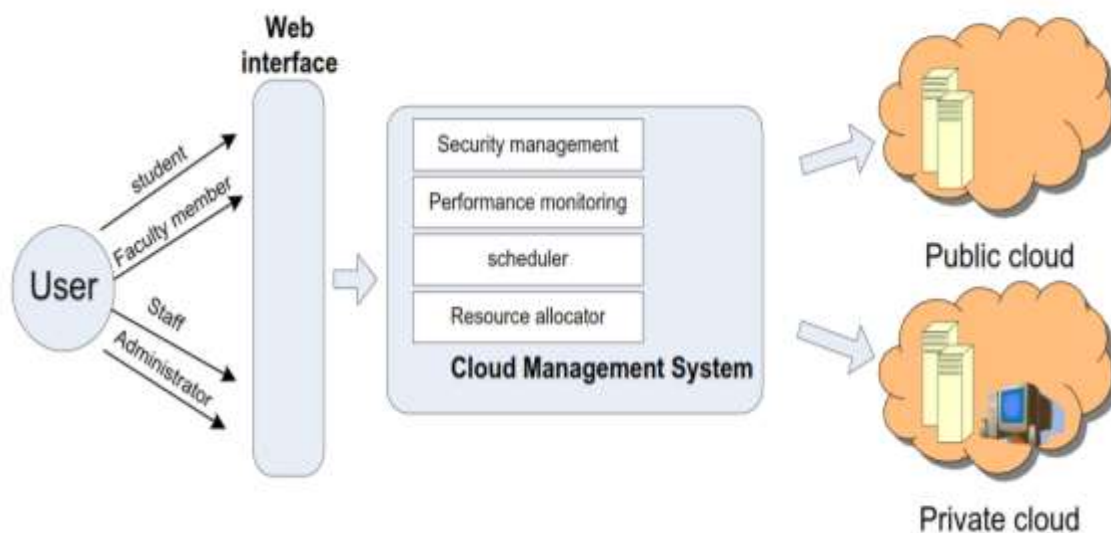


Figure 10: Ucloud Architecture (Sqalli et al., 2012)

2.8 Experimental Model on Curbing the Weaknesses of IDS and Firewall on Policy

Violation in Cloud

According to Souley and Abubakar (2018), CAPTCHA is a tool commonly used in IPS, to prevent machine intruders (bots) from intruding into a system, however, in this research work, CAPTCHA was used as IDS. The technique to be used is cognizance of the fact that there are software intruders that can read CAPTCHA and attempt to infiltrate it and intrude into the system. CAPTCHAs with weak design pattern and fixed length with varying

colors on the text was employed for use in web-based system acting like IPS while in real sense it is an IDS that will attempt to lewd software intruders using machine learning-based attack to successfully read the text-based CAPTCHA and infiltrates the system. Likewise, software intruders using unwitting human labor can easily read the CAPTCHAs and infiltrates the system as well. The CAPTCHA character is to not only be read and re-type back to the system, but it is also to be read, understand and abide by. For instance, one of such CAPTCHA may be displayed as “DO NOT TYPE ANYTHING IN THE TEXTBOX BELOW”. A wise human will understand that the textbox should be left empty, while a software intruder that successfully reads the words would just rush and type “DO NOT TYPE ANYTHING IN THE TEXTBOX BELOW” in the textbox or something similar to that sentence. As soon as anything is typed in the provided textbox, the system quickly detects that an intrusion has taken place and the intruder is quickly redirected to the honeypot model for post-intrusion activities. Figure 11 illustrates the IDS using CAPTCHA as a trap.

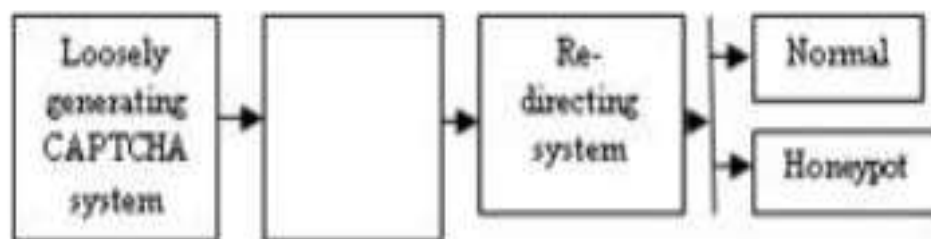


Figure 11: IDS using CAPTCHA as a trap (Souley, & Abubakar, 2018)

Recent cloud-based intrusion detection techniques have predominantly employed the MapReduce model, an abstract programming model that processes large datasets on clusters of computers. MapReduce is composed of two steps; map and reduce. In the MapReduce model, a large dataset is split and each split sent to a node, also known as a mapper, where each split is independently processed. Mapper results are then shuffled, sorted, and passed to reducers that digest and prepare the results (Fontugne et al., 2014).

A possible implementation of MapReduce for a cloud-based intrusion detection technique is inherently simple. The map step examines each split for traffic anomalies, and the reduce step combines them, packages them, and presents the overall report. Many researchers' intrusion detection techniques follow this general approach (Kumar, & Hanumanthappa, 2013). While particulars change, MapReduce remains a foundational tool throughout the most recent research on cloud-based intrusion detection analysis.

2.9 Cloud Security Policy

Cloud policies are the guidelines under which organizations operate in the cloud. The overall cloud computing security strategy is in turn supported by policies, which should clearly explain the necessary compliance and regulatory needs to keep the online cloud environment safe. These policies are documented in every aspect of cloud security, these include; Scope as the specific cloud environments and services that are supposed to be covered, Compliance as the expectations of cloud security in meeting federal, end-users, businesses, and other regulatory requirements. Accountability, which includes the areas and people, is responsible for ensuring a safe cloud-computing environment. Deployment is a high-level view of how cloud security was maintained. Identity and access management, which include who has access to specific information and how identity is authenticated and authorized. Confidentiality and sensitivity is an objective analysis of the confidentiality of specific data sets, applications, and other cloud elements. Acceptable use is the standards that you expect end-users, developers, and other authorized users to abide by and lastly Breach is what happens in the event of a breach of security or policy (Kalaiprasath et al., 2017).

Cloud security policy ensures that the provision of a cloud service in accordance with the business and security requirements and relevant laws and regulations of ISO 27001 (Council-CSCC, 2012).

Table 2: Policy and Baseline Controls (Council-CSCC, 2012)

Cloud computing security policy document	Each organization's cloud computing security policy document: -SHALL be approved by senior management, published and communicated to all employees and relevant external parties either as part of the organization's information security policy or as a separate policy. The policy should set the goals and objectives governing the cloud computing service.
Security program leadership	The senior management responsible for the cloud Computing security policy SHALL be identified by name, title, business phone, business address, and date of designation. Changes to the senior management MUST be documented within thirty (30) calendar days of the effective date of the change.
Review of the Security Policy	The security policy SHALL be reviewed at planned intervals or if significant changes occur to ensure its continuing suitability, adequacy, and effectiveness. For example Architectural changes, service model changes, service upgrades or changing the CSP.
CSP Hardening guides and policies	The organization SHALL assess the CSP virtualization hardening guides and policies and evaluate the party gap assessment against virtualization security standards. This includes but not limited to: Disable or remove all unnecessary interfaces, ports, device, and services; Securely configure all virtual network interfaces and storage areas; Establish limits on VM resource usage; Ensure all operating systems and applications running inside the virtual machine is also hardened; Validate the integrity of the cryptographic key-management operations; Harden individual VM virtual hardware and containers
Snap-Shots Security	The organization SHALL ensure that the CSP has the controls in place to guarantee that only authorized snapshots are taken, and that these snapshots' level of classification and storage location and encryption is compatible with strength with the production virtualization environment.

2.9.1 Security Goals for Cloud Computing

Cloud computing security includes identity and access management, data security, privacy protection, and virtualization security. The network physical hardware and node physical hardware form flexible computing, storage, and network bandwidth through multiple layers of virtualization. The integrated virtual resource pools provide a resource sharing, distribution, management, and control platform with the need to choose. Virtual technology could construct a scalable service-oriented IT infrastructure providing cloud-computing services (Shawish, & Salama, 2014).

For example, Amazon EC2 (elastic compute cloud) provides users with a large number of virtual resources and completes the user tasks through these resources. Users simply create a virtual machine instance according to their needs. Most experts believe that the biggest difference between the cloud and traditional IT environment is its virtual computing environment, which makes security problems become very tricky. Identity Management and Data security issues could be solved through existing access control policies, data encryption, and other traditional security means. The security of the cloud is not inseparable from cloud monitoring technology (Saswade et al., 2016). Virtualized cloud computing as the most important technology and the virtual cloud-computing environment is a unique environment. Virtual machine monitoring (Zou et al., 2013) is a new area of monitoring. Traditional security measures are difficult to fundamentally solve the problems. So new security policies must be adopted (Fatema et al., 2014).

According to Zissis and Lekkas (2012), there are a lot of monitoring tools of cloud; virtualization-based security monitoring is one of them. It uses isolation to monitor and protect a specific system. Therefore, from the standpoint of technology of safety monitoring, research work based virtualization security monitoring can be divided into two categories: internal monitoring and external monitoring.

2.10 Hail marry Attack in Armitage

The Hail Marry Attack is an exploiting option available in Armitage, a Java-based client for Metasploit. Essentially, running every potential exploit you think may work on the system with a single click. Hail Mary feature is a smart DB-autopwn. It finds exploit relevant to your targets, filters the exploit using known information and then sorts them into an optimal order (Ridho, 2014).

2.11 Software Development Models

The software development life cycle or SDLC for short is a methodology for designing, building, and maintaining information and industrial systems. So far, there exist many SDLC models, such as the Waterfall model, which comprises five phases to be completed sequentially in order to develop a software solution; another model called the Spiral model, which is visualized as a process passing through some number of iterations. Finally, the incremental model is any combination of both iterative design or iterative method and incremental building model for software development. It has seven phases, and they are as follows Planning, requirements, analysis, implementation, deployment, testing, and evaluation (Larman, & Basili, 2003).

2.11.1 Waterfall Model

Considered as the traditional method of explaining the software development process in software engineering, the waterfall model happens to clarify the process into a linear flow with a specified sequence to let the users understand that further level is made progressive on completion of the previous one. The Waterfall Model is the oldest and most well-known SDLC model. This model is widely used in government projects and in many major companies. The special feature of this model is its sequential steps. It goes down through the

phases of requirements analysis, design, coding, testing, and maintenance. Moreover, it ensures the design flaws before the development of a product.

This model works well for projects in which quality control is a major concern because of its intensive documentation and planning (Alshamrani, & Bahattab, 2015). Stages that construct this model are not overlapping stages, which means that the waterfall model begins and ends one stage before starting the next one. The following steps give a brief description of the waterfall processes.

Step 1 is a requirement, which is the description of system behavior to be developed. Usually, it is the information provided by clients. Hence, it establishes the agreement between the clients and the developers for the software specifications and features. In short, requirements are gathered, analyzed and then proper documentation is prepared, which helps further in the development process.”In the High-Level design stage, the gathered information from the previous phase is evaluated and proper implementation is formulated. It is the process of planning and problem solving for a software solution. It deals with choosing the appropriate algorithm design, software architecture design, database conceptual schema, logical diagram design, and data structure definition. Furthermore, in the coding stage, the whole requirements will be converted to the production environment. Additionally, the testing stage phase deals with the real testing and checking of the software solutions that have been developed to meet the original requirements. In addition, it is the phase where the bugs and system glitches are found, fixed up, and refined. Lastly, the Maintenance stage after the software is already released; it may need some modifications, improvements, error correction, and refinement accordingly. Thus, this phase is the process of taking care of such concerns (Munassar, & Govardhan, 2010).

2.11.2 Rapid Application Development

Rapid Application Development aimed at providing quick results. Rapid application development is meant to give excellent development processes with the assistance of other development approaches. It is created to take the maximum advantage from the development software. Undoubtedly, it is designed to supplement the workability of the whole software development procedure for highlighting the participation of an active user. Hiring for this kind of development is not straight forward, because there are many factors which you need to take into consideration.

Table 3: Strength and Weaknesses of SDLC Models (Alshamrani, & Bahattab, 2015)

Model /feature	Strengths	Weaknesses	When to Use
Waterfall	<ul style="list-style-type: none"> i. Easy to understand and implement. ii. Widely used and known. iii. Define before design, and design before coding. iv. Being a linear model, It is very simple to implement. v. Works well on mature products and provides structure to inexperienced teams. vi. Minimizes planning overhead vii. Phases are processed and completed one at a time. 	<ul style="list-style-type: none"> i. All requirements must be known upfront ii. Inflexible. iii. Backing up to solve mistakes are difficult, once an application is in the testing stage, it is very difficult to go back and change something that was not well- thought out in the concept stage iv. A non-documentation deliverable only vi. Produced at the final phase. vii. The client may not be clear about what they want and what is needed. ix. Customers may have little opportunity to preview the system until it may be too late. xi. It is not a preferred model for complex and xii. Object-oriented projects. xiii. High amounts of risk and uncertainty, thus, small changes or errors that arise in the completed the software may cause a lot of problems 	<ul style="list-style-type: none"> i. When quality is more important than cost or schedule ii. When requirements are very well known, clear and fixed. iii. The new version of existing iv. The product is needed. v. Porting an existing product to a new platform
Rapid Application Development	<ul style="list-style-type: none"> i. Increased speed of development and increased quality ii. RAD increases quality through the involvement of the user in the analysis 	<ul style="list-style-type: none"> i. Reduced Scalability, ii. Reduced features iii. Reduced scalability occurs because of a RAD 	

- and design stages.
- iii. Advantages of interoperability
- iv. Extensibility
- v. Portability

2.12 Gaps Identified from Previous Studies

Ford et al. (2016) developed an adaptive enterprise IDPS free open-source break-in prevention software. Fail2ban is used to create a data collection agent. Here, all software agents are interconnected to the central behavior analysis database service, collect and record attack meta-information during prior attack attempts. The agents use both real-time and previous data by applying integrating rules from the information analysis method into intrusion prevention policies. However, this proposed system has a high false-positive rate.

According to Xiong (2014), present OpenFlow linked works like OpenNetMon and OpenSafe both proposed the network monitoring service in a cloud environment to efficiently collect the traffic usage statistics and detect malicious activities. However, they do not propose a comprehensive solution for a cloud system. These works did not go beyond the stage of detection and are not able to provide further analysis and countermeasures for any attack. The ‘detecting and alerting’ nature of monitoring solutions demand the human-in-the-loop to inspect the generated security alerts and manually take actions, which cannot respond to attacks in a prompt fashion.

According to Merlo et al. (2016), an adaptive mechanism that considers full account prediction errors and residual traffic was proposed. This model was evaluated using a network simulator and delays were calculated. The results indicated that only a minimal delay is introduced, owing to the security analysis. However, this model lacks an ideal prediction algorithm thus it produces packet delay for false prediction.

Yevdokymenko (2016) designed an adaptive method to detect and prevent active attacks in telecommunication systems. Nevertheless, this approach was not capable to detect

new attacks. Keshri et al. (2016) presented a Denial of Service (DoS) prevention technique using a firewall and IDS based on a mining technique, which included data selection, data preprocessing, transformation, and model selection and evaluation. They used the NSL-KDD dataset, a superior version of the KDD99 cup dataset, for estimation. Though, the technique could not detect intranet attacks.

Banerjee et al., (2018) proposed an Intrusion Filtration System (IFS), which provides strong security and the capability to terminate the execution and distributing of corrupted files. The system can be used offline and provides high throughput. In the approach, all files available in the system are checked, in the sense that the system log is scanned and information about all applications and software installed in the system is stored in the IFS database. The normal way of updating of the database is designed to terminate the dissemination of corrupted files. However, there is no real-world implementation of IFS.

Substantial research has been conducted in this field to defeat the current security threats in cloud computing by implement IDS in the cloud. This is liable for monitoring the utilization of resources for the virtual machine using data acquired from virtual machine monitors (Edwards, & Dalton, 2014). Specifically, all monitoring operations are done outside the virtual machines so the attacker cannot modify the system in cases of breach. However, this method can detect normal activity as intrusion even if it is authorized activity. Consequently, all these breaches should be taken into account during the implementations of IDS within the cloud environment. On the other hand, many other researchers are interested in distributing the IDS among the nodes of the grid within Cloud computing environment in order to monitor each node and alert the other nodes when an attack occurs.

2.13 Conceptual Framework

The conceptual framework for this study was developed by incorporating key variables derived from a review of the research literature on policy violations in the cloud. VM1 is a

virtual machine sign up to get the right privileges to access the model. After the login module, the servers are initiated to allow the server to connect to the network. If the connection takes too long then the VM is taken back to the login module. After connection to the network, it then scans the network for any opened ports and blocks it. Otherwise, access to the application is allowed. The application accessed is then checked for policy violations that can be an intrusion. If no threats detected then the connection is established in the cloud, otherwise, if the threats are detected then it is blocked before the session is created in the cloud.

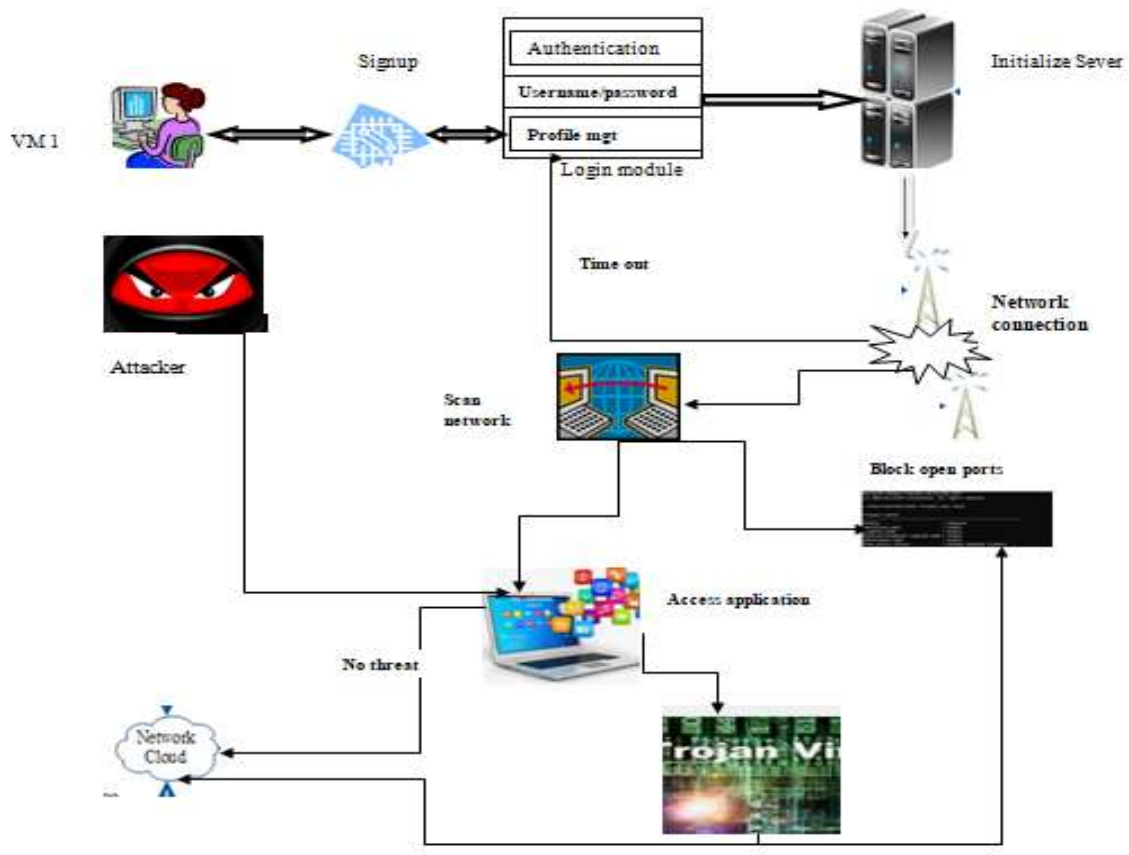


Figure 12: Conceptual Framework (Author, 2019)

CHAPTER THREE

RESEARCH DESIGN AND METHODOLOGY

3.1 Introduction

This chapter presents the research design, prototype development, and model development. It further, entails model evaluation and ethical issues. While conducting this research the focus here is on the development of artifacts with evidential value.

3.2 Research Design

The research design for this study was based on the proof of concept design. Proof of concept method has the advantages of using multiple ways to explore a research problem. Here the researcher used experimental design and a survey of literature for gathering requirements and development of the model. In this research, objective one was achieved by identifying the survey of the literature. Objective two demonstrated the weaknesses of the survey of the literature. Objective three was achieved through Rapid Application development. Rapid Application Development is a development model that prioritizes rapid prototyping and quick feedback over long drawn out development and testing cycles. With rapid application development, developers can make multiple iterations and updates to the software rapidly without needing to start a development schedule from scratch each time. It comprised four stages; Identification of weaknesses on IDS and firewall, Prototype development, model development, and Model evaluation as depicted in figure 13. Objective four was achieved through evaluation. In evaluation, there are criteria used, these are validity, correctness, consistency, and completeness.

There are various artifacts that can potentially be recovered in the Cloud, including VMs, browser artifacts, and network traffic and application logs. For this research, the artifact considered is VMs. This is because they are the most common type of data that users

are able to create in Virtual box. Logs were also considered because they record information on VM ownership and can, therefore, be used to identify the user who created them. Finally, Virtual box, a Linux-based Cloud technology, was selected. This was because the basic specification for a virtual box host is easy to meet in a laboratory environment, where it can be set up for experimental purposes.

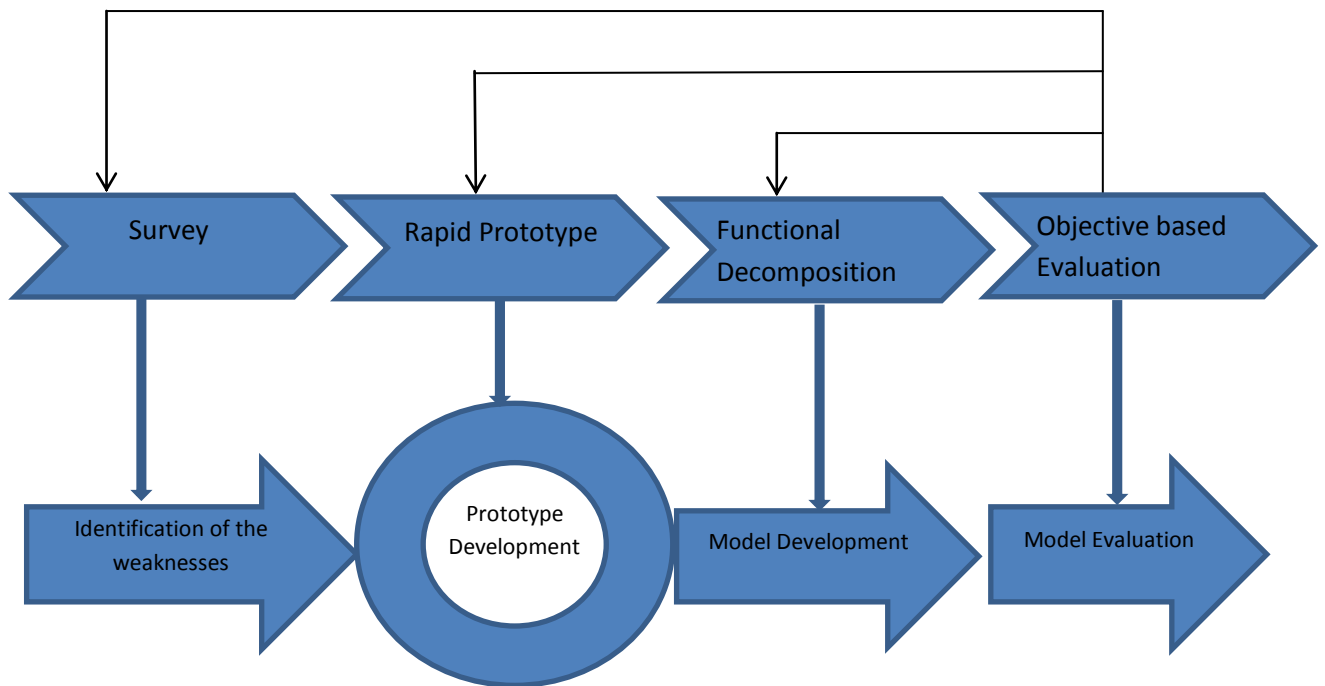


Figure 13: Rapid Application Development

3.3 Identification of the Weaknesses of IDS and Firewall

Every piece of software or application is built for a reason. Rapid Application Development starts with identifying out what project is supposed to accomplish. The researcher started by identifying the weaknesses of IDS, IPS, and firewall through conducting a survey of the literature. In addition, the research expanded on different problems that have been identified by the different researchers in related studies. Based on the literature review a

set of requirements to be fulfilled by the POVIDE model was generated. The objective of the survey of the literature was to identify different weaknesses of IDS, IPS, and firewall.

3.4 Prototype Development

A prototype is a set of rules and methods that describe the functionality, organization, and implementation of the system. The prototype demonstrated various aspects of the product such as interfaces or functionality. The prototype was used as a ‘proof of concept’ in order to aid in the development of a product or model where no clear approach is evident. The prototype was used to look at the approach that would work and demonstrate to a user what the intended model would look like, what it would do and how it would work. The rapid application prototyping approach as depicted in figure 13 used in the translation of a model for detecting IT infrastructure policy violations in the cloud environment.

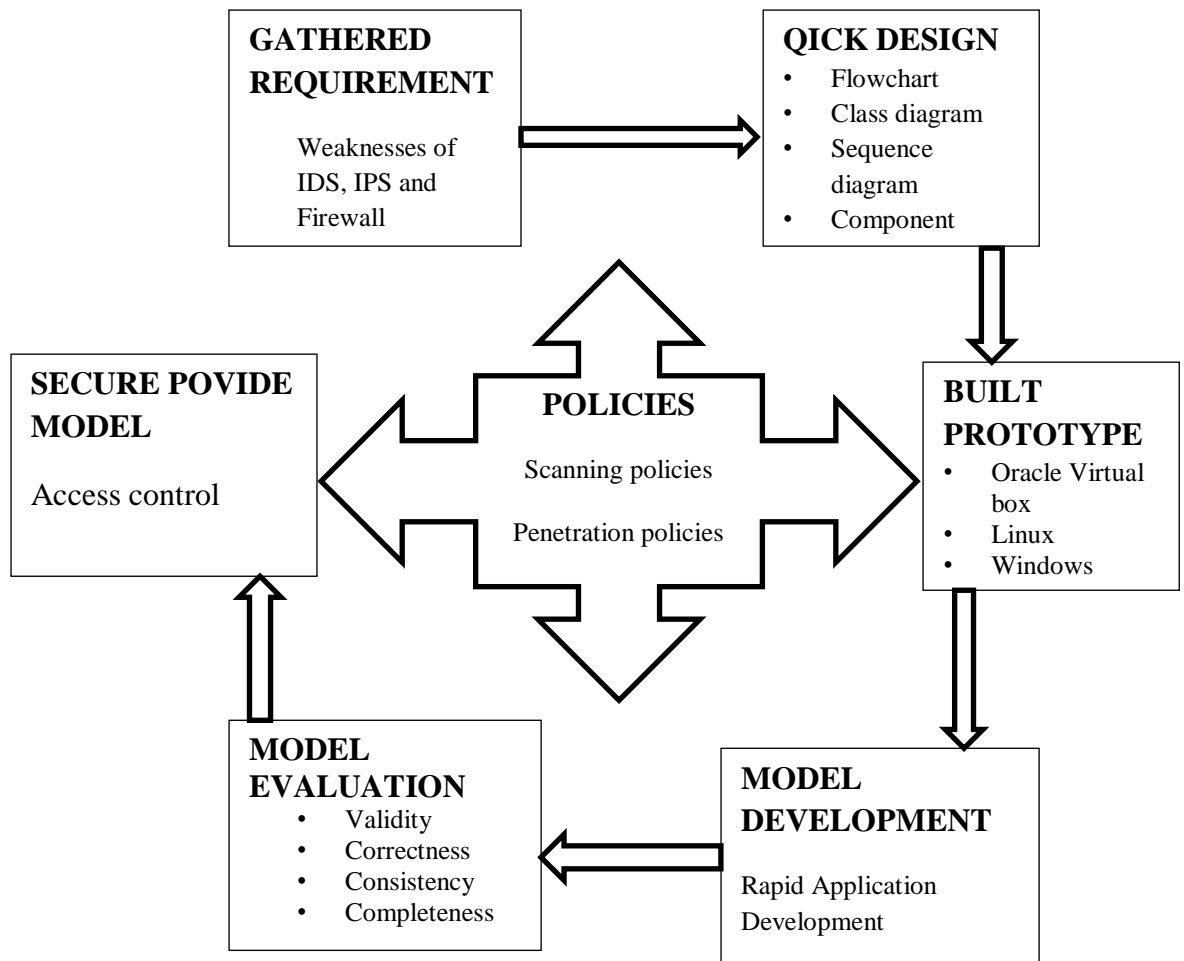


Figure 14: Rapid Prototyping (Author, 2019)

Gathered requirements: The requirements were gathered after identifying the weaknesses of IDS and firewall have been inferred from literature. Quick design: The model processes described was translated into flow chart, class diagram, sequence diagram and component diagram in chapter four. The file system was then designed based on the data that was being stored.

Build prototype: A prototype was then built using Oracle virtual box, which is open-source, and develop using Linux and windows, the file system was used as the database and hosted on the virtual box as part of the development process. The POVIDE model was tested then acquired to test functionality.

Model Development: The system requirements were refined on an ongoing basis using feedback from the model development, deployment, the survey collected and the testing process. The model was developed using the Rapid Application Development process. **Model testing:** Once the requirements are satisfied, a final version of the model was completed and tested with real users in a pilot study.

POVIDE model: Feedback from this stage informed any further development and refinement of the POVIDE security to make it more secure. The access control was one of the main features to make the model secure. **Policies:** all the steps followed were guided by the policies outline and put in place to enable the POVIDE model to be secure so that users cannot violate the policies in place. The policies adhered to were scanning policies and penetration policies.

In this stage, a functional model was built to demonstrate the weakness of IDS, IPS, and firewall. This would help in curbing policy violations in the cloud. In terms of this research, it intends to use three routers to represent the different networks. Router one was ufw (uncomplicated firewall); Router two and three is Snort. It is hosted on Oracle VM, a virtual box manager that requires the users to start Oracle VM. The external attacker uses the Kali Linux pen test while internal attackers use Linux 2.6/3x/4x (64bits).

The first experiment demonstrated the weaknesses of the firewall. The research demonstrated weaknesses of firewalls on rapid prototype development. Router one was the firewall, there was a virtual machine, an external machine to represent outside attacker and an internal attacker is also represented with one machine. All these machines were connected to the firewall network. Here the network used Ubuntu (64bits). The external and internal machines used Kali Linux 2.6/3x/4x (64bits). The other virtual machine used windows. The next experiment was to test whether the machine represented as an external attacker could

access the Virtual Machine on the network and exploit it. It also tested whether the internal attacker could scan the machines on the network.

The second experiment demonstrated the weaknesses of IDS and IPS on rapid prototype development. A snort was considered for router two. There were virtual machines, external machines and internal machines connected to the network. The experiment tested false positive and false negative. Router2- IDS connects to the gitbash. The experiment demonstrated that the virtual machine would access the website while gitbash that is connected to snort would report it as a threat thus demonstrating to false positive alert. The snort indicated that it required human intervention. Machines used are windows that are connected to servers that use the Linux Ubuntu version.

3.5 PROVIDE Model Development

POVIDE model uses an Oracle VM which is a free and open source server virtualization and has a virtual box manager. The virtual box manager kernel is installed to operate as a standalone server POVIDE model is made up Pf sense firewall and snort. The model used a Red Hat Linux version that is connected to the Pf sense firewall and snort. The model used a machine that is hypervisor (a VM monitor), which is a Type-1 or native hypervisor that runs directly on the host's hardware, as opposed to a Type 2 hypervisor which runs on the host's OS. The model hypervisor enables the running of multiple instances of an OS on a single host as well as the multiple OS concurrently on a single host, on the same hardware. The experiment indicated that VMs with the Windows OS installed are the most widely used. POVIDE model can be deployed with either local or shared storage, where 'shared' refers to the storage being shared in a pool of provided model hosts. However, both were used in this research in order to understand their impact on artifact recovery. This means that all VMs created will be stored on a local disk, which will allow easier access to them.

VMs stored on local storage cannot be migrated between hosts. The model intends to use a shared network file system (NFS). Finally, the POVIDE model uses scanning and penetration for testing. In scanning, an intense scanning was tested to see which ports are open. The results of the scanning process were in two virtual machines. The second stage used the Hail Mary attack that is Penetration stage. It was done by sending all the exploits from the external attacker computer to the server computer through open ports. The rapid prototyping approach depicted in figure 13 was used in the translation Policy Violation Detection model into a functional prototype.

3.6 Network Diagram

One of the characteristics of Cloud computing is that it offers broad network access. Cloud resources are available over a network that can be internal or external, local or the Internet. As a Cloud technology, the POVIDE model needs a network interface before it can be installed, but it does not need the Internet to work. While the basic network configuration for the POVIDE model is a Local Area Network (LAN), a Personal Area Network (PAN) works as well (Chaudhary et al., 2014). A PAN is considered a subset of a LAN and can connect to other networks, including the Internet (Gilchrist, 2016). In terms of this research, a LAN was used in order to provide a controlled environment where a management system could be connected to the same network as the POVIDE model hosts. In the network diagram, there is a router1, which is a firewall. The firewall network is connected to three host computers. One is represented as an internal attacker. Then there is a different computer that is not on the network to represent outside attackers (external attackers). The second and third are the network that is IDS and IPS respectively. They are connected to three computers each. The last network is the POVIDE model network. Different computers on the network

have different IP addresses and they are connected to different servers. The basic layout is shown in figure 15.

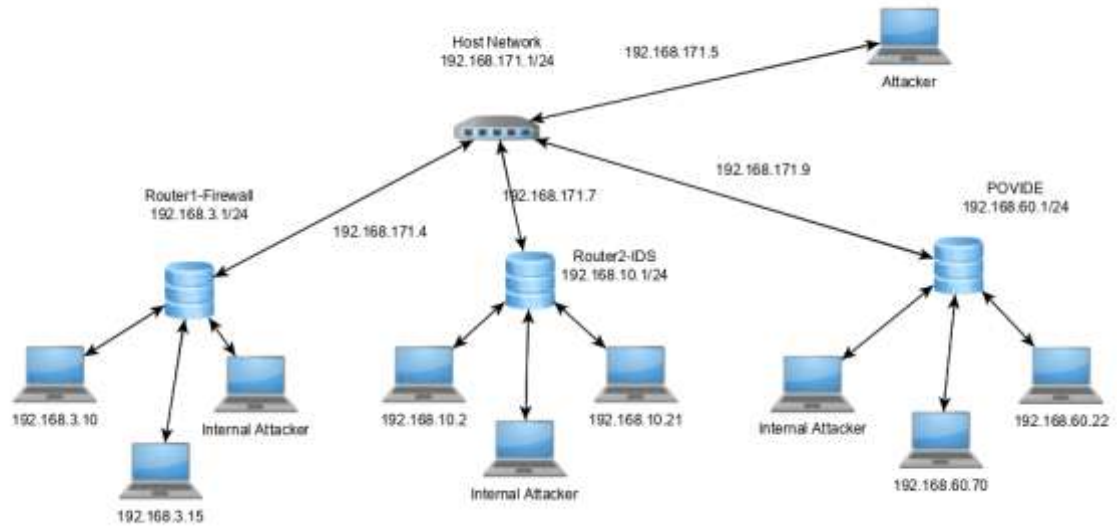


Figure 15: Network Diagram (Author, 2019)

3.7 Population of the Study

The targeted populations were people working in ICT. Senior network administrators of eight local universities based in Nakuru County and Safaricom did the testing. Network administrators of local universities represented a larger group of different universities in Kenya and Safaricom is one of the biggest organizations in Kenya. Senior network administrators represented a sample group to come up with the different institutions and organizations as they represented the universities and organizations in Kenya as shown in table 4.

Table 4: Organizations and Institutions (Author, 2019)

Organizations/ Institutions	Number of Network Administrators
Kabarak University	1
Egerton University	1
Jomo Kenyatta University of Agriculture and Technology	1
Kenyatta University	1
St. Paul University	1
Mount Kenya University	1
University of Nairobi	1
Laikipia University	1
Safaricom	2
TOTAL	10

3.8 Data Analysis

For ten respondents that assessed the model, their responses were on a Likert scale. They responded with ‘agree’, ‘disagree’ and ‘neutral’. In a contingency table, the counts in each cell were assumed independent and had a Poisson distribution. A Poisson regression analysis in Log Linear Model was carried out with the response variable being a count, the distribution was Poisson and the link function was logarithm; i.e $\eta = \log(\mu)$. The Data was analyzed on the SAS package.

3.9 Reliability and Validity of the Instrument

Reliability is the measure of the degree to which research yields consistent results or data after repeated trials. It is the degree of consistency used by research instruments or procedures for demonstration. Poor reliability degrades the precision of a single measurement and reduces the ability to track changes in measurement in a study (Mugenda and Mugenda 2003). A reliable instrument consistently produces the expected results when used more than once to collect data from the same subjects randomly drawn from the population (Mugenda and Mugenda 2003). The data obtained from a pilot study was used to estimate the reliability of the instrument. Cronbach's alpha coefficient was used to estimate the reliability of the evaluation forms. This is because all the three instruments were rated based on scales with a range of scores. The instruments should return the Cronbach's reliability coefficient of at least 0.7 in order to be accepted reliable. From the reliability analysis, it can be observed that the value of Cronbach's alpha is 0.718, which is greater than 0.7 alpha. This means that the scale conforms to internal reliability. This means that the researcher can use the tool for evaluating expert opinion.

Table 5: Reliability and Validity Test Results

Cronbach's Alpha	N of Items
.718	11

3.10 Model Evaluation

A formative evaluation was used. Senior network administrators who are experts on the model tested the prototype. Usability testing was used to allow for a small number of individuals to try the model out and give feedback.

The criteria used in the PROVIDE model are validity, correctness, consistency, and completeness. In terms of validity, it should be possible to show that the origin of the proof, as well as the processes and circumstances that produced weaknesses, cannot easily be

disputed. In terms of correctness, the machine that created the proof should be in proper working condition and the techniques that were used to process the weaknesses of existing tools should be acceptable within the context of the study. Consistency requires justifiable methods to obtain and process the proof. Finally, completeness demonstrates that the maximum amount of tests required for the study to be collected and analyzed.

Table 6: Evaluation Procedure (Author, 2019)

Criteria	Evaluation
Validity	<ul style="list-style-type: none"> • The circumstance that produced Weaknesses • Processes • Authenticity
Correctness	<ul style="list-style-type: none"> • Proper working condition • Results generated should be the same • Scanning and blocking of ports and files • Accuracy
Consistency	<ul style="list-style-type: none"> • Reliability • Scanning the network • Penetration to the network
Completeness	<ul style="list-style-type: none"> • Audit log • Performance of the model • Short response time

3.11 Research Authorization

As per academic requirements in Kenya, a research permit and an introduction letter were obtained from the National Commission for Science, Technology, Innovation, and Communication (NACOSTI), before embarking on the model development. They are presented in appendix III and IV.

3.12 Ethical Considerations

Ethics refers to acceptable behavior while conducting research. The researcher is expected to avoid harm to anyone and to resolve any potential conflicts with integrity. As such, ethics is concerned with ensuring that all research participants are protected and

promoting values such as trust, accountability, mutual respect, and fairness. Informed consent is one of the means by which a participant's right to autonomy is protected. It is the ability for self-determination in action according to a personal plan. Informed consent seeks to incorporate the rights of autonomous individuals through self-determination.

The issue of confidentiality and anonymity is closely connected with the rights of beneficence, respect for dignity and fidelity. Anonymity is protected when the subject's identity cannot be linked with personal responses. If the researcher is not able to promise anonymity, he has to address confidentiality, which is the management of private information by the researcher in order to protect the subject's identity. The individuals who tested the model their results were made confidential. It is necessary to have skills and knowledge for the specific investigation to be carried out and be aware of the limits of personal competence in research. Any lack of knowledge in the area under research must be clearly stated. Inexperienced participants should work under qualified supervision, which has to be reviewed by an ethics committee.

The participants were assured that their participation would be treated in confidence and that data collected would be used for academic purposes only. The research did not include any information deemed to threaten the security of participants. However, a limitation of this is that the validity of the results might have been endangered.

CHAPTER FOUR

DATA ANALYSIS, PRESENTATION, AND DISCUSSIONS

4.1 Introduction

This chapter presents the interpretation of the findings of the study as set out in the research methodology. It focuses on testing of the Model and elements of the data security implementation for cloud computing in the organizations. The research data was gathered exclusively through a survey of the literature review as the secondary research instrument. The weaknesses of IDS and Firewall were designed in line with the research objectives of the study. The purpose of the study was to develop Model to detect policy violation that is resilient, secure and capable to curb policy violation. The model was developed using Rapid application development and was taken through a proof of concept. It was then tested through validity, correctness, consistency, and completeness.

4.2 Analysis

This section presents the results of experiments that were set up to verify the weaknesses of the Firewall and IDS and to document the structure of the POVIDE model. The purpose of the experiment was to find out how a good POVIDE model detects network packets as normal or attack data. In this research, system experiments are performed by the virtual machine host acting as an attacker by using several types of attacks against multiple hosts that act as targets of attack, and then observed and analyzed the accuracy of the system in recognizing this type of attack.

The results are presented as each stage of the experiment is reported. The system that was set up for this set of experiments was composed of windows 10, Ubuntu 64 bit, 100 GB HDD and 8 GB RAM. A partition editor was then installed using the Ubuntu Software

Centre. In one network it used at least One GB memory. Storage is a file system that has an IDE controller and SATA.

There are two stages of testing. The first stage is intense scan all TCP ports. The second stage is the Hail Mary attack. Research results, which are obtained after demonstrating the weaknesses of the firewall and IDS and testing PROVIDE Model, are discussed in the next section.

4.3 Weaknesses of IDS and Firewall

The advent of the Internet, personal computers and computer networks are becoming increasingly vulnerable to various kinds of attacks. The attacks are usually caused by a failure to implement strong security policies and failure of using of security tool that is strong. The various security tools that are available are Firewall, Intrusion Detection System and Intrusion Prevention System. Each tool has its own features, strength, and weaknesses. Some of the weaknesses are presented in table 7.

Table 7:Weaknesses of IDS, IPS, and Firewall (Author, 2019)

Detection and Prevention tools	Weaknesses
IDS/IPS	<ul style="list-style-type: none">i.IDS is not an alternative to strong user identification and authentication mechanism (Chowdhary et al.,2014).ii. not a solution to all security concerns (Patel et al., 2013).iii. Network Administrator is required to examine the attack once it is detected and reported.iv. False positives occur when IDS incorrectly identifies normal activity as being malicious (Ford et al., 2016).v. False negatives occur when IDS fails to detect malicious activity (Latha, 2016)
Firewall	<ul style="list-style-type: none">i. Firewalls use a set of laws that are physically configured to differentiate legitimate traffic from non-legitimate traffic (Hock, & Kortis, 2015).ii. The firewall cannot counter a network attack nor can it start effective counter-measures.iii. Most firewalls do not evaluate the contents of the data packets that build up network traffic (Eronen, & Zitting, 2001).iv. Firewalls cannot avert attacks coming from Intranet (Bensefia, & Ghoualm, 2011).v. Filtering rules of the firewall cannot prevent attack coming from the application layer (Nurika et al., 2012)

4.3.1. Demonstration of the existing Weaknesses of the Firewall

In the firewall demonstration, the weaknesses that were tested are external and internal attacks. The external attacker can hack on the computer on the network that has a firewall and an internal computer can scan the network within and without the firewall detection.

4.3.2. Experimental Setup for the Firewall

This section presents the result of the experiment that was undertaken in order to verify the findings of the survey of the literature in relation to the firewall. This process was then used to document firewall prevention tools.

The first experiment was carried out using three virtual machines as shown in figure 16. The firewall server was installed on the first system with 20GB HDD and 2GB RAM

using the default settings and static network settings. After the firewall was installed, the disk was viewed in the Ubuntu partition of an analysis machine with the automatic mount option enabled.

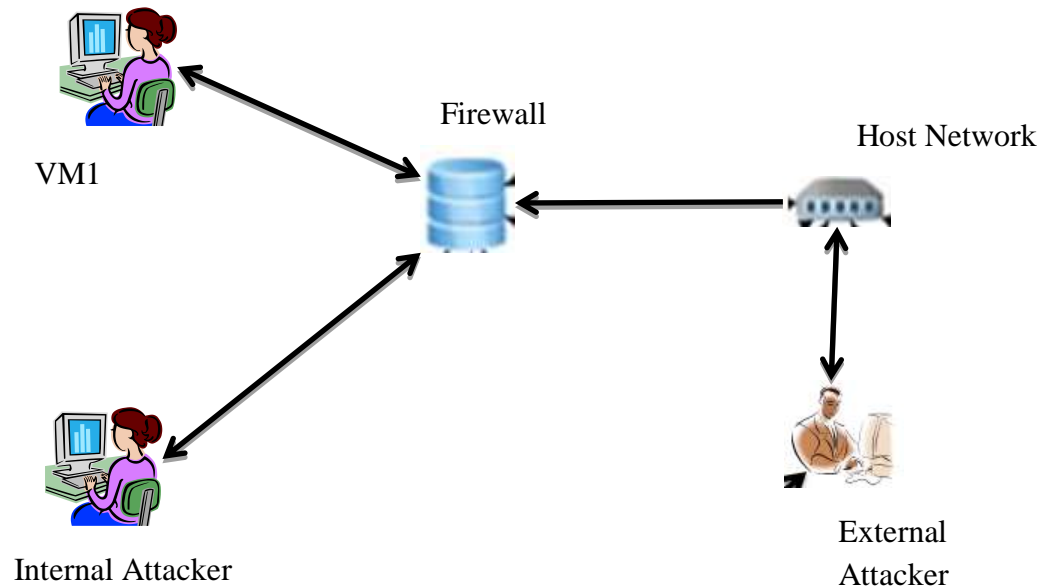


Figure 16: Firewall Setup (Author, 2019)

An internal attacker scans the entire network for an open network in order to attack. External attacker hacks to the system on the network by sending an exploit in form of download to VM1, and it manages to download the file thus enabling the external attacker from getting a screenshot on what application VM1 is accessing.

4.3.3 Algorithm for Demonstrating Weaknesses of Firewall

Step 1: Start

Step 2 User: User request to be signed up in the firewall server

Step 3 Actors: Verifies and add the user to the server

Step 4: Firewall server prompt user to enter username and password

a. The user enters username and password

b. Server checks username

If username does not exist in the file system it sends invalid username to go to step 2

Step 5 Short descriptions: Login verifies user access to firewall server based on username and password pair that is entered by the user on the end-user device (personal computers or iPad or smartphone)

Step 6 User initializes the devices used in the server at the same time

Step 7 Pre-condition: User is connected to the network (tested through IP address ping with VM1, external attacker, internal attacker, and the firewall server)

Step 8 Invariant: If time out occurs, restarts the procedures go to step 4 above

Step 9: External attacker exploits VM1 by sending a download file

Step 10: VM1 downloads the file that contains exploit without knowing

If the external attacker can access and get a screenshot of what VM1 is accessing, then go to step 12 else go to step 13

Step 11: Internal attacker to scan and access the entire network without exploit (scan target 192.168.3.0/24). If the internal attacker can scan and access the network then go to step 12 else go to step 13

Step 12 Weakness has been demonstrated

Step 13 Stop, connection established in cloud

4.3.4 Flowchart Diagram for Firewall

The main functionality of the firewall system is to receive an incoming and outgoing connection from the network and virtual machines. The network administrator adds the virtual machines on the network. The virtual machines accessing the network must have a valid credential for signing in to the network. The device being used is initialized. In response to a connection request initiating a connection between respective endpoints in the

network and virtual machines. First, the attacker from the outside network sends an exploit to the virtual machines in terms of the file, the virtual machine then downloads the file or not. The firewall server then performs analysis, if the firewall detects the exploit then the connection is established in the cloud else the weakness has been demonstrated. On the other hand, the VM2, which is, also an internal attacker scans the network for open ports in order to attack. This is because the firewall cannot prevent attacks coming from Intranet. If it can scan the network then it means the weakness has been demonstrated else, the connection is established on the network.

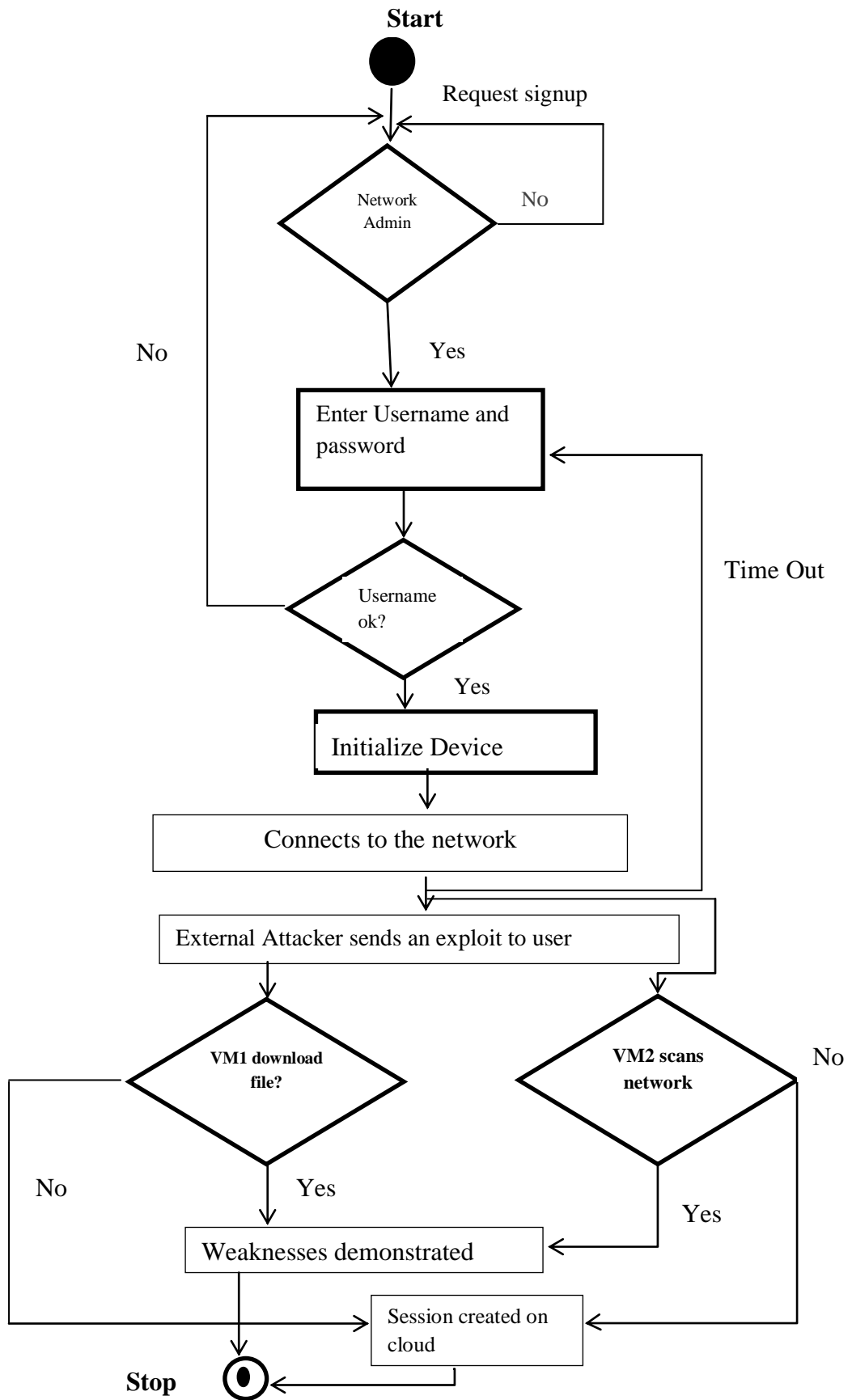


Figure 17: Flowchart Diagram for Firewall (Author, 2019)

4.3.5 Physical Volume Usage

This shows the type operating system used and the usage of the memory, processor and hard disk. Figure 18 shows that the firewall below used Ubuntu version 18.04.2 LTS. The hard disk used was 10 GB, the firewall requires 23 percent of the memory.

```
$ ssh -l ruth -p 14607 localhost
ruth@localhost's password:
Welcome to Ubuntu 18.04.2 LTS (GNU/Linux 4.15.0-45-generic x86_64)

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:       https://ubuntu.com/advantage

System information as of Sat Apr 27 13:17:10 UTC 2019

System load:  0.0          Users logged in:  1
Usage of /:   37.7% of 9.78GB  IP address for enp0s3: 10.0.2.15
Memory usage: 23%          IP address for enp0s8: 192.168.3.3
Swap usage:  0%           IP address for enp0s9: 192.168.171.4
Processes:   89

* Ubuntu's Kubernetes 1.14 distributions can bypass Docker and use containerd
  directly, see https://bit.ly/ubuntu-containerd or try it now with
```

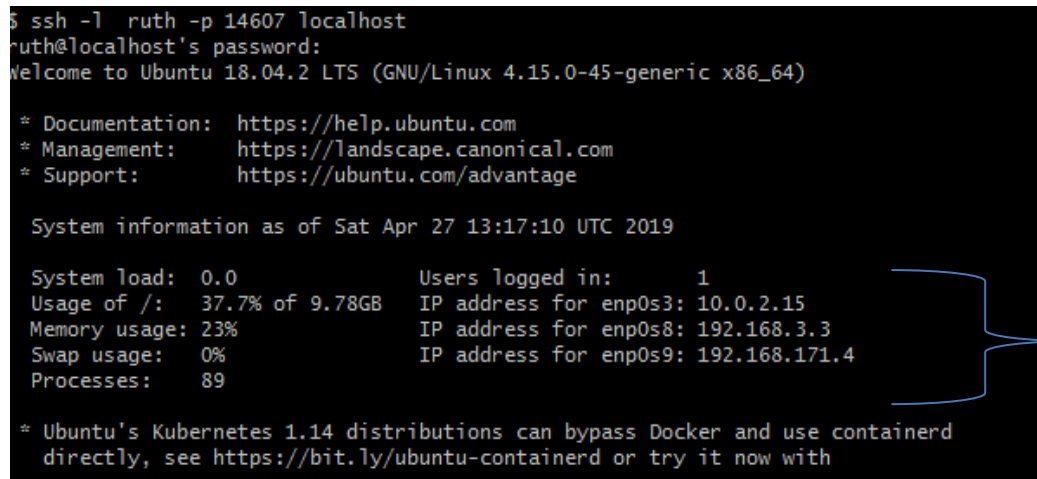


Figure 18: Physical Volume Usage (Author, 2019)

The `ssh -l ruth -p 14607 localhost` command was then used to securely operate network services over an unsecured network. Next `ruth@localhost`' password was used to display the metadata on any physical volume on the system, as shown in figure 18. The version for the Ubuntu used is 18.04.2 LTS.

4.3.6. Login Credentials

Figure 19 shows the login image for the firewall. The first part shows an incorrect user name and password. After using correct credentials it then shows the requirement of the server for it to work properly. It gives the amount of memory to be used and the storage size. It also shows the network address used and the IP address.

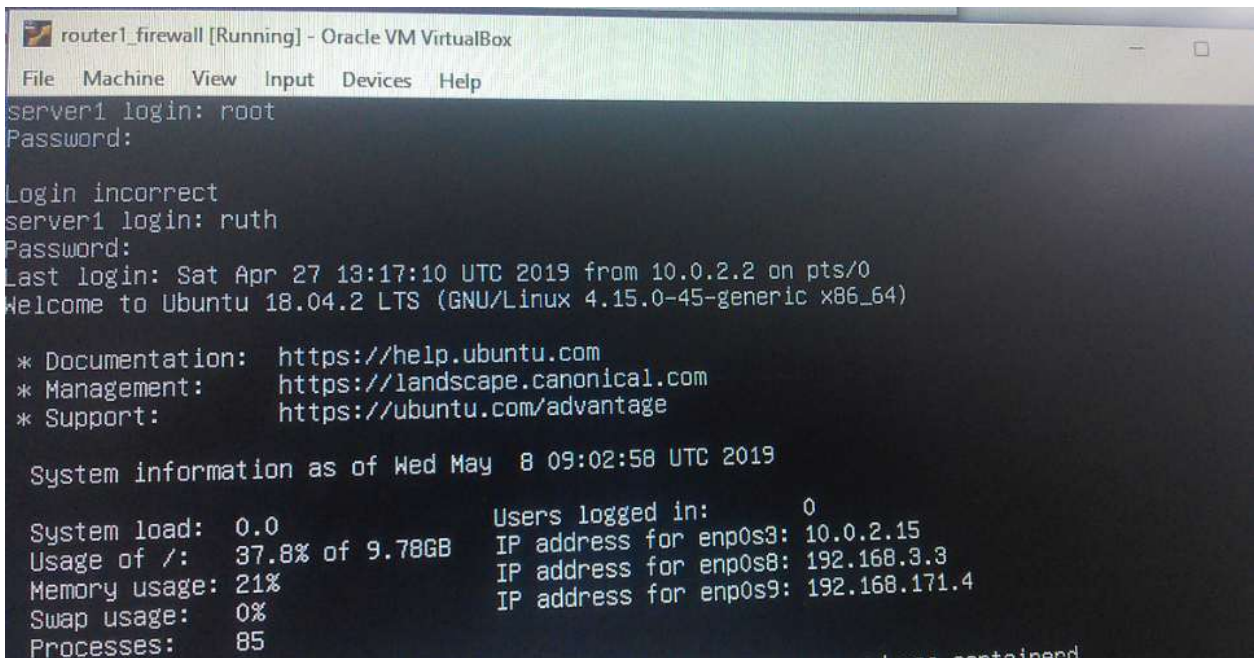


Figure 19: Login Credentials (Author, 2019)

4.3.6.1 Incorrect Logging

This is the logging credentials required for the user to successfully access the firewall server. The user requires a valid username and password given to them by the network administrator. Before login, the user needs to sign up through the network administrator.

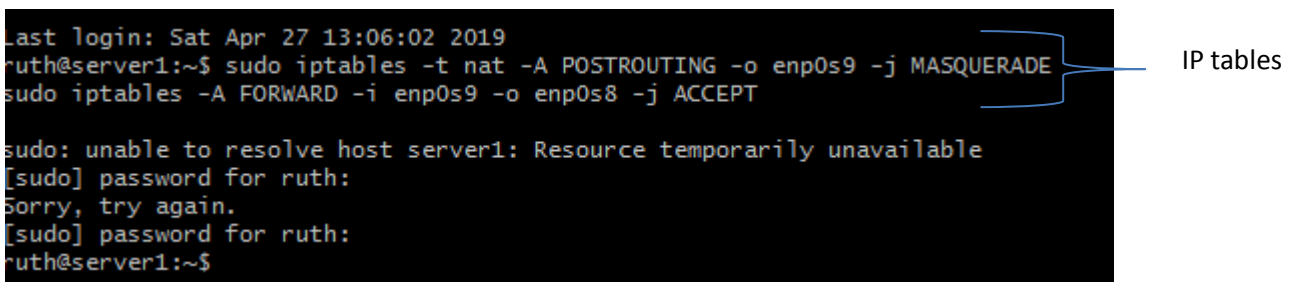


Figure 20: Incorrect Logging (Author, 2019)

IP tables command was used to allow a system administrator to configure the tables provided by the Linux kernel firewall and the rules it stores. ACCEPT was used to view the contents of IP tables and it revealed that the metadata of the POSTROUTING was attached after that of the IP tables was configured. IP tables are to initialize the device used to the network. The correct credentials for username and password must be used. Figure 20 required

the correct login in order to access the network using a valid username and password. The valid user is ruth@server1 as shown in figure 20. It also shows the date and time of the last login.

4.3.6.2 Connection to the Network

Figure 4.6 shows the connection established on the network. This means that different virtual machines can communicate with each other and communicate to the server and outside attacker's virtual machine.

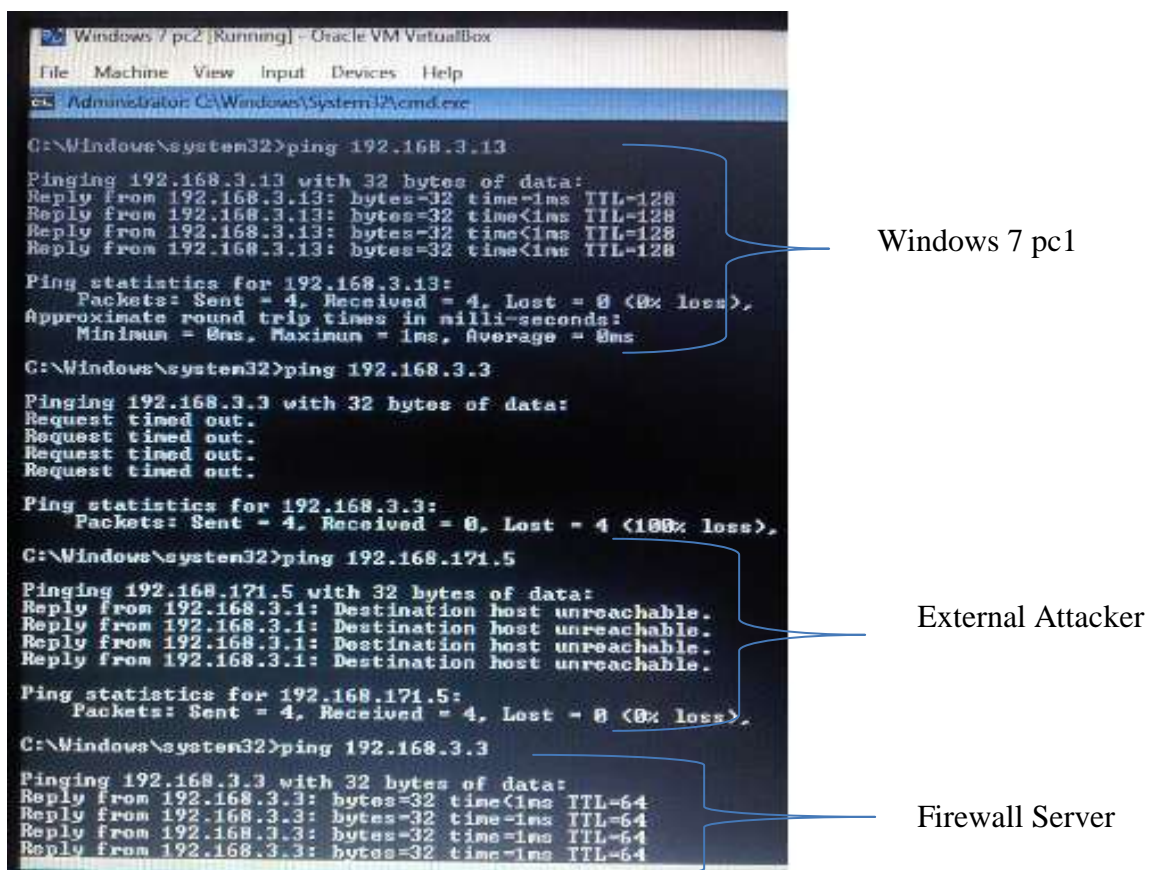


Figure 21: Connection to the network (Author, 2019)

The commands in figure 21 show communication from different hosts on the network. The results show that there is a communication between the machines with the firewall server and an external attacker. The first connection was the window machine1 with an IP address of 192.168.3.1 to communicate to the other virtual machine with an IP address of

192.168.3.13. It shows very clearly that there are packets sent and received respectively. Ping 192.168.3.13 shows that the user pc can communicate with other virtual machines on the network. The virtual machine with IP address of 192.168.3.13 is presumed by the researcher to be the internal attacker.

The second connection was to communicate with the server. Here the researcher disabled the firewall to test the connection to the server. The results showed that there was no communication with the server. It indicated request timeout, to illustrate no connection. The firewall was then enabled and the connection of IP address 192.168.3.3 was established. Ping 192.168.3.3 was then used to connect to the server.

The third connection was to communicate with the virtual machine outside the network. This virtual machine is presumed to be an external attacker. This shows the virtual machine can be communicated to the external attacker whose IP address is 192.168.171.5. Figure 21 shows packets sent and received by the virtual machine to the external machine.

4.3.7 Penetration Stage of Firewall

Penetration was done by sending all the exploits from the attacker's computer to the server computer through open ports. Open ports are obtained from the scanning stage. The delivery exploits have the objective to do penetration. Penetration was conducted in order to find the weakness of a firewall server. Exploits are delivered automatically adjusted by Armitage.

4.3.7.1 Penetration of Exploit

The command below shows the exploit used by the external attacker with an IP address of 192.168.171.5 to attack any virtual machine on the network. The external pc launches an attack through an open port of 443. Metasploit has 1863 exploits that can be used

to attack a machine. The exploit launch dialog to allow the user to configure options for a module and choose whether to use a reverse connect payload. If you launch an individual client-side exploit, you have the option of customizing the payload that goes with it. In a penetration test, it is usually easy to get someone to run your evil package. The hard part is to get past network devices that limit outgoing traffic. For these situations, it helps to know about meterpreter's payload communication options. There are payloads that speak HTTP, HTTPS, and even communicate to IPv6 hosts. A payload handler is a server that runs in Metasploit. Its job is to wait for a payload to connect to your Metasploit and establish a session. The multi/handler output is used to create a handler that waits for the payload to connect.

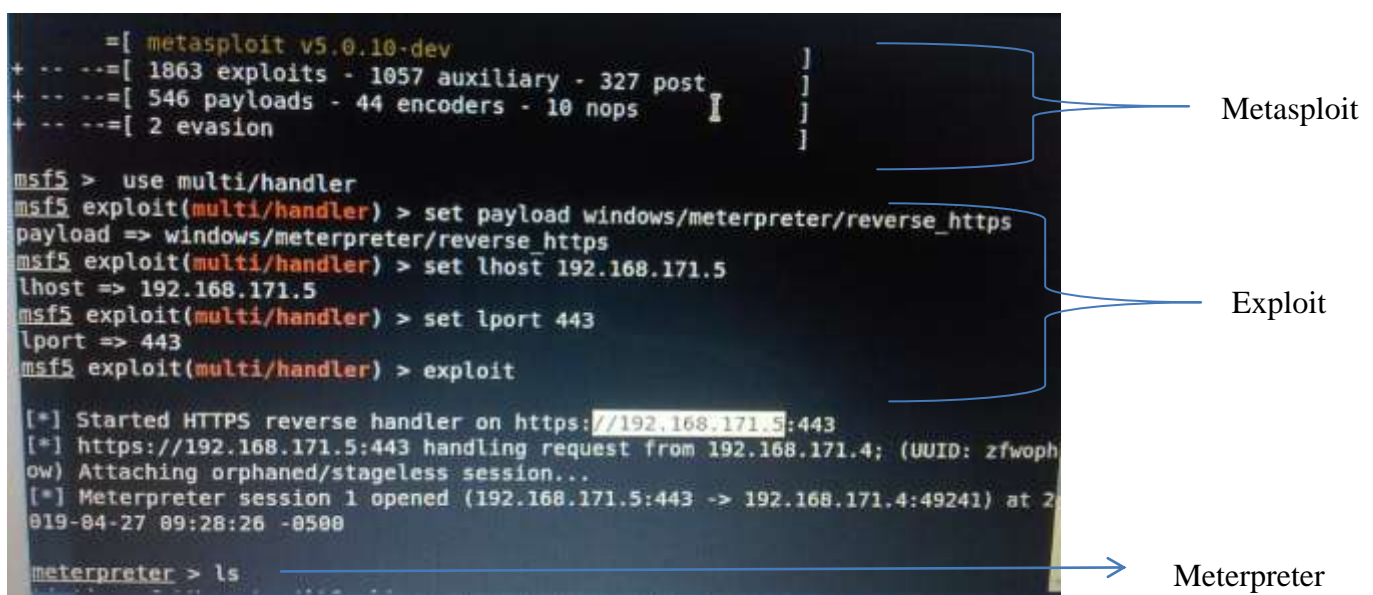


Figure 22: Penetration of Exploit (Author, 2019)

Metasploit's RPC daemon and the Armitage team server are not GUI programs. You may run these over SSH. Penetration testers will find this feature invaluable. Imagine you are working on a pen test and come across a system you do not know much about. You can reach back to your company and ask your local expert to load Armitage and connect to the same Metasploit instance. They immediately have access to your scan data and they can interact with your existing sessions seamlessly. Some Metasploit modules require you to specify one

or more files. If a file option has next to it, then you may double-click that option name to choose a local file to use. Metasploit shell sessions are automatically locked and unlocked when in use. If another user is interacting with a shell, Armitage will warn you that it is in use. Armitage will upload the chosen local file and set the option to its remote location for you. Generally, Armitage moves files between you and the shared Metasploit server to create the illusion that you are using Metasploit locally.

The Metasploit command was used as a hacking tool to access the pcs on the network. Next, the exploit command was the processes of the external attacker gaining access to the desktop machines that are on the network. It was used to allow the user to send their IP address to the hacker. The results were as expected the external attacker gained access through meterpreter as shown in the figure above. Meterpreter command shows that the external attacker of IP address 192.168.171.5 has gained access to the user computer host. The msf5 command was used to scan for a handful of open ports. It then enumerates several common services using Metasploit auxiliary modules built for the purpose. The results in figure 4.7 show one session has been opened using port 443. This shows that an exploit has been launched.

4.3.7.2 Exploited file

Figure 23 shows an example of a file that has been downloaded but contains an exploit. Here the user of the virtual machine has downloaded a file, and the attacker can access his or her machine.

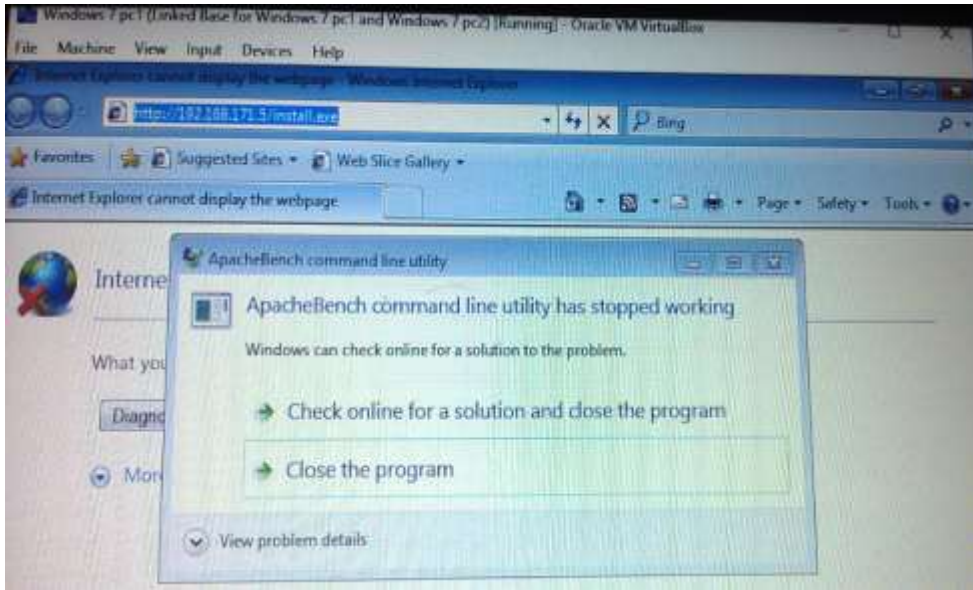


Figure 23: Exploited File (Author, 2019)

The screenshot above shows the user of VM1 accessing a download file from the internet that is presumed to be an external attacker. The file has been downloaded from <http://192.168.171.5/install.exe> check the URL above. As a user, if you download and run the file, it makes the external attacker with an IP address of 192.168.171.5 to access your machine as shown in figure 23.

4.3.7.3 Exploited Virtual Machine

Figure 24 shows a machine that has been captured by an attacker and screenshot has been taken on what the virtual machine is accessing. Meterpreter shows that a session has been created between the virtual machine and an attacker. It shows that the external attacker can access all the files on the machine desktop by showing the URL below on `c:\users\ruth\desktop`. The attacker can also access the size of the file, last modified date, and name of the disk where the file was saved on.

```
meterpreter > ls
Listing: C:\Users\ruth\Desktop
=====
Mode                Size           Type             Last modified          Name
----                -
100666/rw-rw-rw-  1331685      fil             2019-04-27 16:56:34 -0500  Document.rtf
100666/rw-rw-rw-   282          fil             2019-02-15 13:28:39 -0600  desktop.ini

meterpreter > screenshot
Screenshot saved to: /root/LQWqtJZc.jpeg
meterpreter > |
```

Figure 24: Exploited Virtual Machine (Author, 2019)

ls command shows the partitions and the files that an attacker can access the user's machine. The results were as expected, as shown in figure 24. The /root/ LQWqtJZc.jpeg directory was viewed and it was found that the information of the file of this directory corresponded to the file on the external attacker's home page as shown in the figure below. Screenshot command is used to access what the user is doing in real-time. It also shows where the screenshot has stored.

4.3.7.4 Saved Captured File

Figure 25 shows where the attacker stores the attached file from different machines. All the files were captured during testing. The lastly captured screenshot file is LQWqtZc.jpeg.

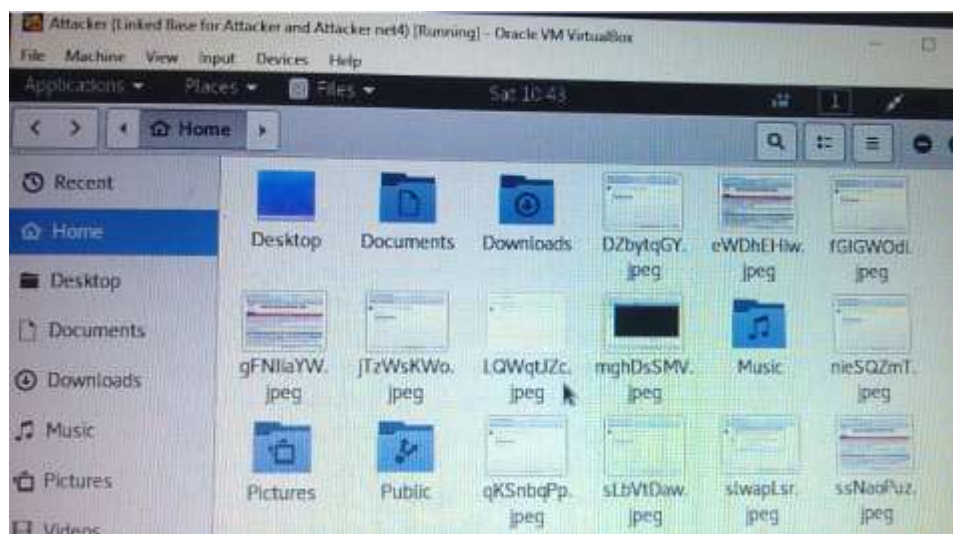


Figure 25: Saved Captured file (Author, 2019)

In addition to this, the /root/LQWqtJZc.jpeg directory was viewed and it was found that the information in the metadata file of this directory corresponded to the metadata on the windows7 machine1. The metadata file is assigned name as a picture. The results were as expected, as shown in Figure 25 that shows where the attacker has saved the information accessed from the user's machine. Figure 25 is the desktop of the attacker where the captured file is stored.

4.3.7.5 The Screenshot File

To confirm the captured file as the right one, the file was opened. It was found out to be the screenshot captured during the users downloaded the file. Figure 26 shows the results of the captured file. It shows the file name, properties of file and file size. It shows the directory where the file was saved in the attacker's machine.

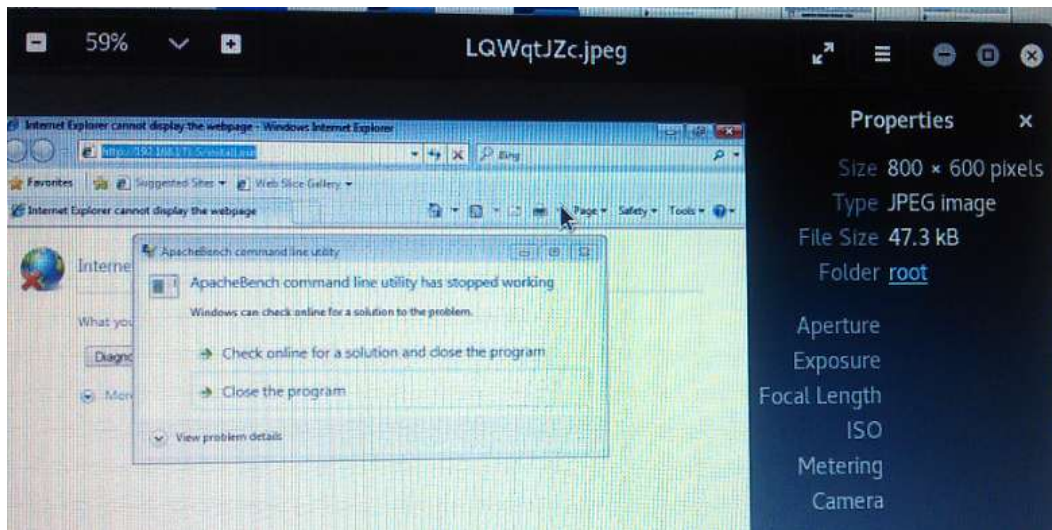


Figure 26: Screenshot File (Author, 2019)

The screenshot shows the external attacker has gained access to the user's machine and has a screenshot of what he or she is accessing on the machine. The results were as expected, as shown in Figure 26. This shows that the firewall could not detect unknown attacks. The same sentiment was echoed by Jansen and Tanner (2014), who stated that an

operation is mostly based on signatures for known-bad software, so systems are not ready for day-zero exploits for which signatures are not yet available. In addition, the placement of these security mechanisms on the local machine allows malware, after a successful compromise, to disable the security mechanisms or hide from them. To curb this weakness the open ports should be blocked and train users not to download a file from unknown sources. There should be a model to detect unknown attacks. PROVIDE Model would block open ports and detect the rogue file from an attacker. The Model would also check on attack response rules and used it to detect an attack.

4.3.8 Scanning Stage of Firewall

The purpose of scanning is to see which port is open. The results of the scanning process are in two computers. The router with the IP Address 192.168.3.3 is the network server where the firewall server application is installed. The computer with the IP address 192.168.3.13 is the virtual machine for the internal attacker, the internal attacker VM is used for scanning. The user virtual machine has the IP address of 192.168.3.1. Open ports are port 22, 445 and 135. Open ports are used for exploits. This would be curbed by blocking the ports and machines with open ports until the user contacts the network administrator.

4.3.8.1 Scanning by the Internal Attacker

Attacker net1 is a virtual machine on the network. Here the internal attacker can scan the network as shown in figure 4.12. The exploiting tool used for scanning was Zenmap as shown in figure 4.13 and 4.14 respectively. Intense scanning is a compressive scanning of the network. It scans all TCP ports. While a scan is running and after it completes, the output of the nmap command is shown on the screen.

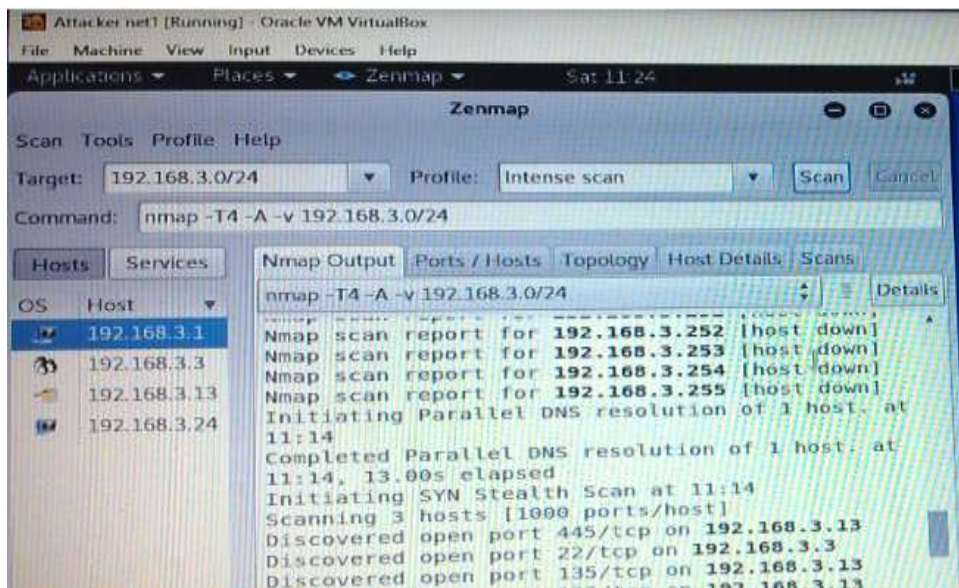


Figure 27: Scanning by Internal Network (Author, 2019)

The `nmap-T4-A-V 192.168.3.0/24` command was used to scan the entire network server for any opened ports by the internal attacker. Then the Nmap command was used to scan and report the IP addresses. The results were as expected, as shown in Figure above. The figure also showed that `192.168.3/24` was used to view the content of the whole network server and scanning details. After scanning the network four host IP addresses were found to be opened, namely `192.168.3.1`, `192.168.3.3`, `192.168.3.13` and `192.168.3.24`. Next, NMAP output was used to view the metadata and display the information.

There were four hosts' machines with open ports such as 445, 22, and 135, which was on the network server. Nmap scans do not use the pivots you have set up. This means that the firewall could not detect threats from the internal attacker. This test was compared to that of Keshri et al. (2016) where they presented a Denial of Service (DoS) prevention technique using a firewall and based on data mining techniques, which comprises data selection, data preprocessing, transformation, and model selection and evaluation. However, the technique could not detect internal attacks. This weakness could be solved by making sure that all open ports are blocked and denied access. This would allow the internal attacker would not be able to scan the network.

4.3.8.2 Tool used for Scanning

The tool used for scanning was Zenmap found in information gathering as shown in figure 28



Figure 28: Tools used for Scanning

Here the researcher used information gathering and Zenmap was used as the scanning tool to scan the whole network. Zenmap was used because it was easier to configure. As shown the figure 29 below.

4.3.8.3 Zenmap

The internal attacker used Zenmap to scan the entire network. Zenmap is an example of information-gathering tools. It is used as a threat as it can scan the network.



Figure 29: Zenmap

4.4 Weaknesses of Intrusion Detection System

IDS has weak policies such as false positive, false negative and IDS are susceptible to protocol-based attacks. One significant issue with IDS is that it regularly gives an alert of false positive. In many cases, false positives are more frequent than actual threats. False-positive is the process of sending a false alert as a threat when it is actually not a threat. Another common weak policy is a false negative. In many cases IDS does not alert on the actual threats instead it ignores it.

4.4.1 Demonstration of the existing Weaknesses of the Intrusion Detection system

In the IDS demonstration, the weaknesses that were tested are a false positive and false negative. False-positive is when snort used in IDS is sending an alert of attack while the user is accessing a website. A false negative is when an external attacker is able to hack on the user's pc without being detected by the snort used in IDS.

4.4.2 Experimental set up of IDS

The experiments were carried out using two virtual machines. One host network server and IDS server. IDS was installed on the first system with 80 GB HDD and 2GB RAM using the default settings and static network settings.

In many cases IDS does not alert on the actual threats instead it ignores it. The experiment was conducted using two virtual machines as shown in figure 30.

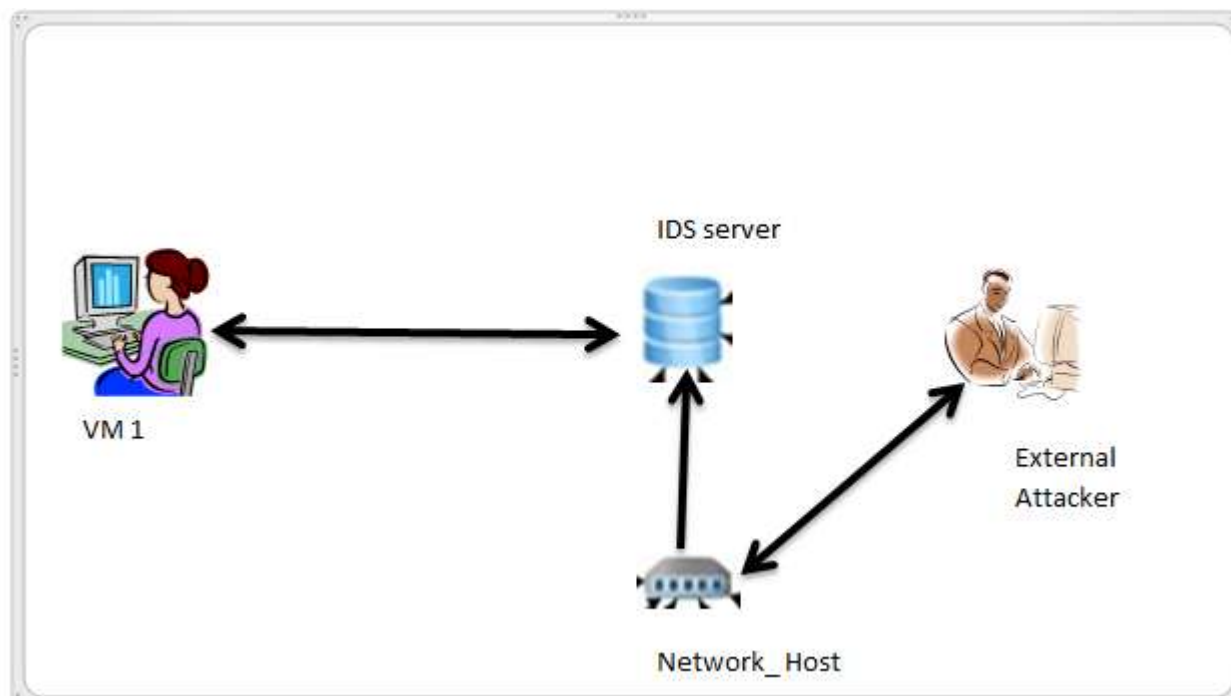


Figure 30: IDS Setup (Author, 2019)

Windows 7 with 64 bits were used as VM. Then `ls` command was used to view the content of the logical volume and directory, a snort was used as the IDS server. In false positive VM1 access a website from the internet while the IDS server reports it as a threat. In the false negative the External attacker is able to launch an exploit that is a threat to VM1 without being detected by the IDS server.

4.4.3 Algorithm for Demonstrating Weaknesses of IDS

Step 1: Start

Step 2 User: User request to be signed up in the IDS server

Step 3 Actors: Verifies and add the user to the server

Step 4: IDS server prompt user to enter username and password

- a. The user enters username and password
- b. Server checks username

If username does not exist in the file system it sends invalid username to go to step 2

Step 5 Short descriptions: Login verifies user access to firewall server based on username and password pair that is entered by the user on the end-user device (personal computers or iPad or smartphone)

Step 6 User initializes the devices used in the server at the same time

Step 7 Pre-condition: User is connected to the network (tested through IP address ping with VM1, external attacker and the IDS server)

Step 8 Invariant: If time out occurs, restarts the procedures go to step 3 above

Step 9: VM1 accesses the website on the internet

Step 10: IDS server views the site and sends a report as a threat alert then go to step 12 else go step 13 (false positive)

Step 11: External attacker manages to send an exploit file to VM1. If VM1 downloads the file and IDS server fails to detect it go to step 12 else go to step 13 (false negative)

Step 12 Weakness has been demonstrated

Step 13 Stop, connection established in cloud

4.4.4 Flowchart Diagram for IDS

The main work of the Intrusion Detection System is to demonstrate the false positive and false negative respectively. The IDS was demonstrated using the Snort and virtual machines. The network admin adds the virtual machines on the network. The virtual machines accessing the network must have a valid credential for signing in to the network. The device being used is initialized. In response to a connection request initiating a connection between respective endpoints in the network and virtual machines.

In the first test, VM1 accesses a website on the internet, if the IDS server detects a threat. It sends the report on the threats and that means that false positive is demonstrated. In the second test external machine send an exploit, VM2 downloads, the exploited file if IDS detects a threat then the session is created on the cloud. If the IDS does not detect the threat then false negative is demonstrated. Both the tests end whether the demonstration is achieved or not.

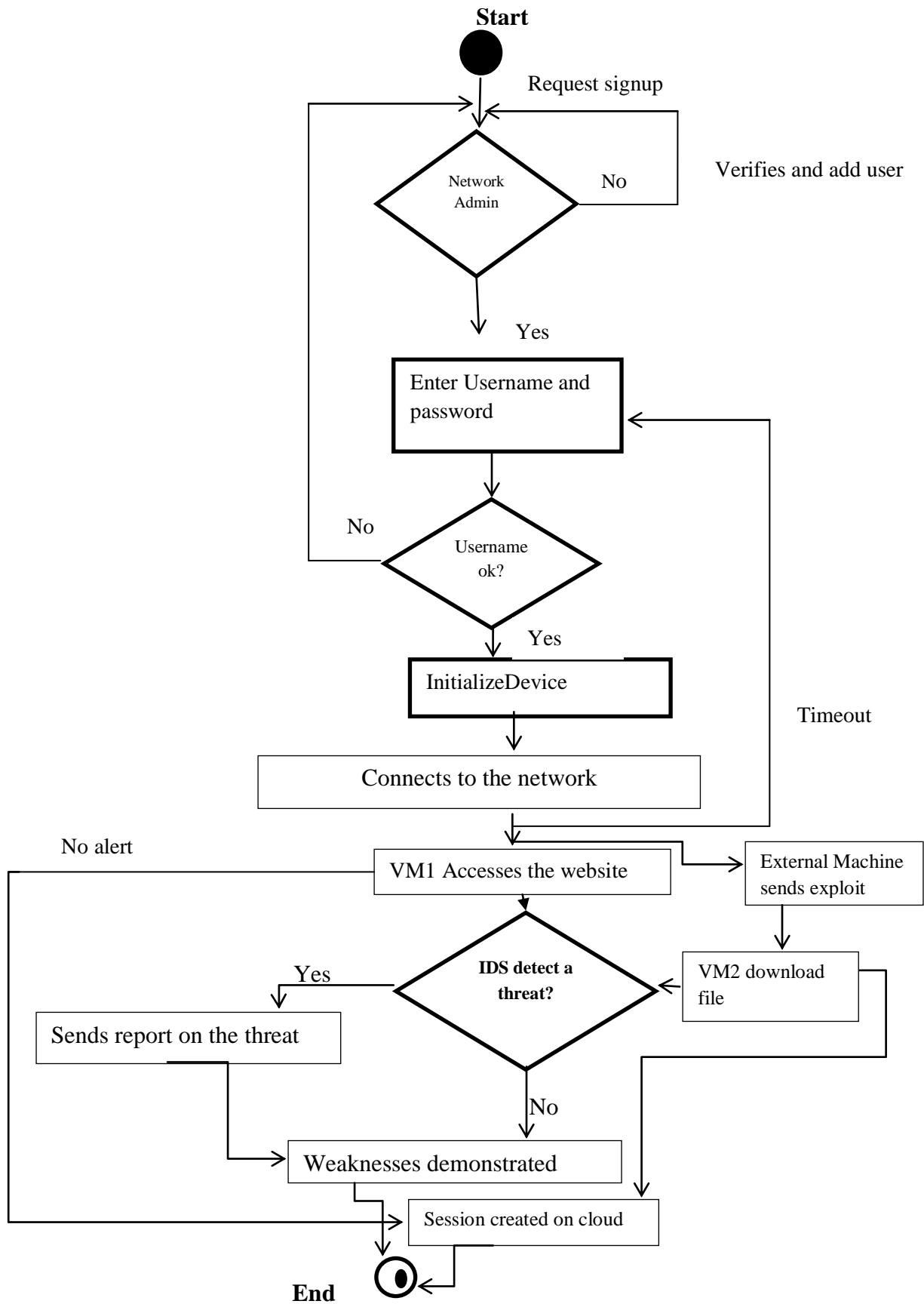
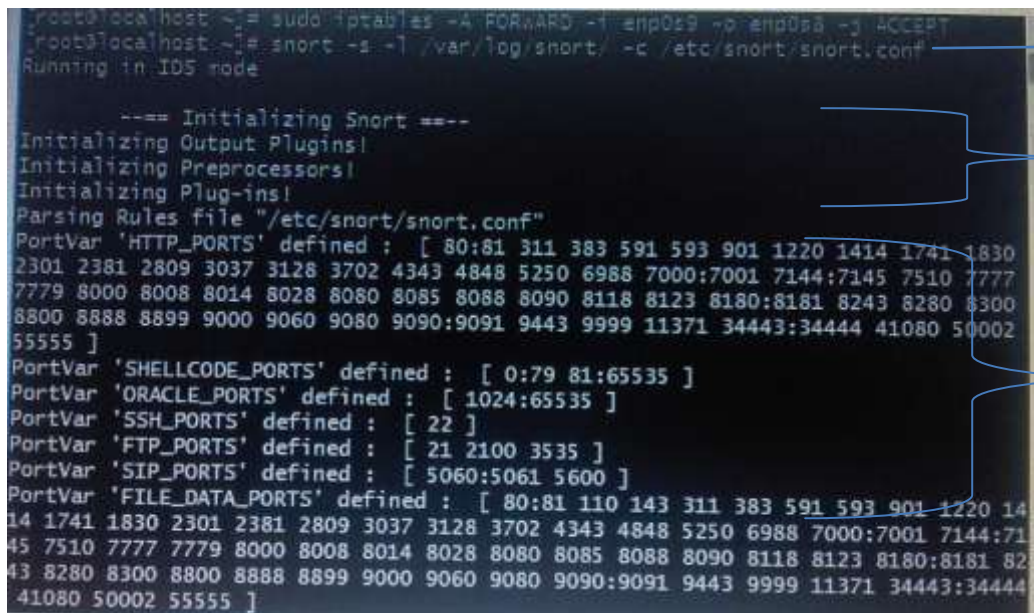


Figure 31: Flowchart Diagram for IDS (Author, 2019)

4.4.5 Snort Configuration

In demonstrating, the false positive Snort was used as an IDS used to test false positive. In figure 32 shows the configuration and initialization of snort. The figure also shows the different ports used by a snort.



```
root@localhost ~# sudo iptables -A FORWARD -i eth0:9 -o eth0:8 -j ACCEPT
root@localhost ~# snort -s -l /var/log/snort/ -c /etc/snort/snort.conf
Running in IDS mode

--- Initializing Snort ---
Initializing Output Plugins!
Initializing Preprocessors!
Initializing Plug-ins!
Parsing Rules file "/etc/snort/snort.conf"
PortVar 'HTTP_PORTS' defined : [ 80:81 311 383 591 593 901 1220 1414 1741 1830
2301 2381 2809 3037 3128 3702 4343 4848 5250 6988 7000:7001 7144:7145 7510 7777
7779 8000 8008 8014 8028 8080 8085 8088 8090 8118 8123 8180:8181 8243 8280 8300
8800 8888 8899 9000 9060 9080 9090:9091 9443 9999 11371 34443:34444 41080 50002
55555 ]
PortVar 'SHELLCODE_PORTS' defined : [ 0:79 81:65535 ]
PortVar 'ORACLE_PORTS' defined : [ 1024:65535 ]
PortVar 'SSH_PORTS' defined : [ 22 ]
PortVar 'FTP_PORTS' defined : [ 21 2100 3535 ]
PortVar 'SIP_PORTS' defined : [ 5060:5061 5600 ]
PortVar 'FILE_DATA_PORTS' defined : [ 80:81 110 143 311 383 591 593 901 1220 14
14 1741 1830 2301 2381 2809 3037 3128 3702 4343 4848 5250 6988 7000:7001 7144:71
45 7510 7777 7779 8000 8008 8014 8028 8080 8085 8088 8090 8118 8123 8180:8181 82
43 8280 8300 8800 8888 8899 9000 9060 9080 9090:9091 9443 9999 11371 34443:34444
41080 50002 55555 ]
```

Annotations in the image:

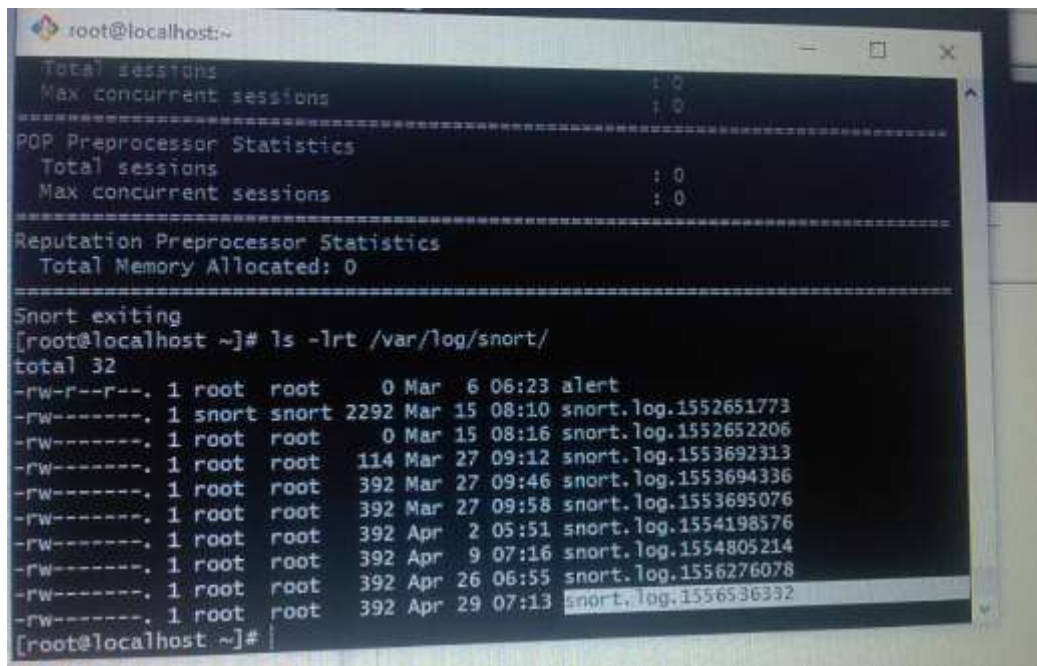
- An arrow points to the command `snort -s -l /var/log/snort/ -c /etc/snort/snort.conf` with the label "Configuration".
- Two brackets on the right side group the lines `--- Initializing Snort ---`, `Initializing Output Plugins!`, `Initializing Preprocessors!`, and `Initializing Plug-ins!` with the label "Initializing snort".
- A bracket on the right side groups the `PortVar` definitions with the label "Http_Ports".

Figure 32: Snort Configuration (Author, 2019)

The `ssh -l root -p 14606 localhost` command was then used to securely operate network services over an unsecured network, here the root was used because the researcher used centos. Next `root@localhost`'s password was used to display the metadata on any physical volume on the system, as shown above. `snort -s -l /var/log/snort/ -c /etc/snort/snort.conf` command used to view the configuration of the snort and to run the IDS mode. It was found out that the IDS was running this as shown in the figure above the first part. The next stage was the initializing snort; this command was used to reset the snort. It was found out that output plugins and preprocessors were reset and configured under the file in the directory called `"/etc/snort/snort.conf"` as shown in the figure above the second the art. Lastly, the `http_` ports were defined it was fund to allow the IDS server to use different ports so longs they are defined and configured. The results were as expected as shown in figure 32.

4.4.5.1 Log file

These are a file found in snort when a virtual machine is accessing the network through the network. These are files sent by a snort as alerts instead of a normal file accessed from the internet. When the snort log is opened it shows an alert while the virtual machine is accessing the website as shown in figure 33



```
root@localhost:~#
Total sessions : 0
Max concurrent sessions : 0
=====
POP Preprocessor Statistics
Total sessions : 0
Max concurrent sessions : 0
=====
Reputation Preprocessor Statistics
Total Memory Allocated: 0
=====
Snort exiting
[root@localhost ~]# ls -lrt /var/log/snort/
total 32
-rw-r--r--. 1 root root 0 Mar 6 06:23 alert
-rw-----. 1 snort snort 2292 Mar 15 08:10 snort.log.1552651773
-rw-----. 1 root root 0 Mar 15 08:16 snort.log.1552652206
-rw-----. 1 root root 114 Mar 27 09:12 snort.log.1553692313
-rw-----. 1 root root 392 Mar 27 09:46 snort.log.1553694336
-rw-----. 1 root root 392 Mar 27 09:58 snort.log.1553695076
-rw-----. 1 root root 392 Apr 2 05:51 snort.log.1554198576
-rw-----. 1 root root 392 Apr 9 07:16 snort.log.1554805214
-rw-----. 1 root root 392 Apr 26 06:55 snort.log.1556276078
-rw-----. 1 root root 392 Apr 29 07:13 snort.log.1556536332
[root@localhost ~]#
```

Figure 33: Log File (Author, 2019)

ls -lrt /var/log/snort/ was a command to view a directory of the log files that the snort is able to write into file directory. ls command views the directory of the log file. It was found out that it only contains two types of files that are alert file and snort.log PCAP (packet capture). Alert file contains alert metadata in a text format. While snort.log PCAP contained packets that triggered alerts as shown in figure 33, Snort.log displays the size of the file, date and time and log number.

4.4.6 False Positive

A false positive is an instance where an IDS incorrectly identifies a normal activity to be malicious. During normal operation, an IDS can generate thousands of false alarms per

day. Network intrusion detection systems no matter if they are anomaly-based or signature-based - share a common problem: the high number of false alerts or false positives. These problems usually cause the user, the network administrator to lose confidence in the alerts, lower the defense levels in order to reduce the number of false positives or to have an overload of work to recognize true attacks due to IDS mistakes.

4.4.6.1 False alert

This is the example of the false alert, while the site accessed is actually a website. The results as shown in figure 34



Figure 34: False Alert (Author, 2019)

Curl <http://www.testmyids.com/> this a website the user's pc is accessing. The website was found to contain data such uid=0(root) gid=0(root) groups=0(root). cat/var/log/snort/snort.log.1556536332 commands read the alert log from snort and gather the list of critical ports to give a report of the file. It was found out that snort gave a report of the live attack. The report also shows the content type, length of the file, date and time accessed. It further gives the server the file is received from and the last time the file was modified. The result was as expected, as shown in figure 34. This result was compared to Ford et al. (2016)

who developed an adaptive enterprise IDS. Free open-source break-in prevention software and Fail2ban used to create the data collection agent. The agents used both real-time and previous data by applying integrated rules from the information analysis method into intrusion prevention policies. However, this proposed system had a high false positive rate. This weakness of false positive can be curbed by developing a strong technique that would only detect an alert when it happened. POVIDE Model would only detect malicious attacks that use open ports. It would also analyze the network.

4.4.6.2 Start Apache

Service apache2 start is used to run the file used for hacking into a user's machine. The result shows that the external attacker has successfully started the hacking file. Service apache was used to run an exploit. For the exploit to work in snort then service apache must be running. Apache2 is a service script used to start/stop/ restart the Apache2 service under Debian or Ubuntu Linux. You need to log in as root or use sudo command restart Apache as shown in figure 35

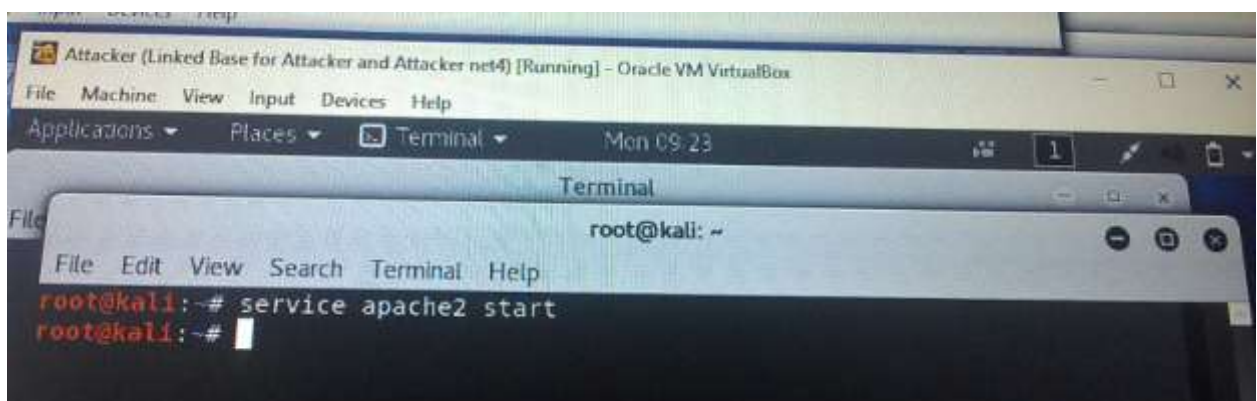


Figure 35: Start Apache2 (Author, 2019)

4.4.6.3 Exploiting Tools

There are different exploiting tools that you can use, in this research the exploiting tool was used to allow penetration. This was to test the false negative. In this case, the exploiting tool used was Metasploit as shown in figures 36 and 37 respectively.

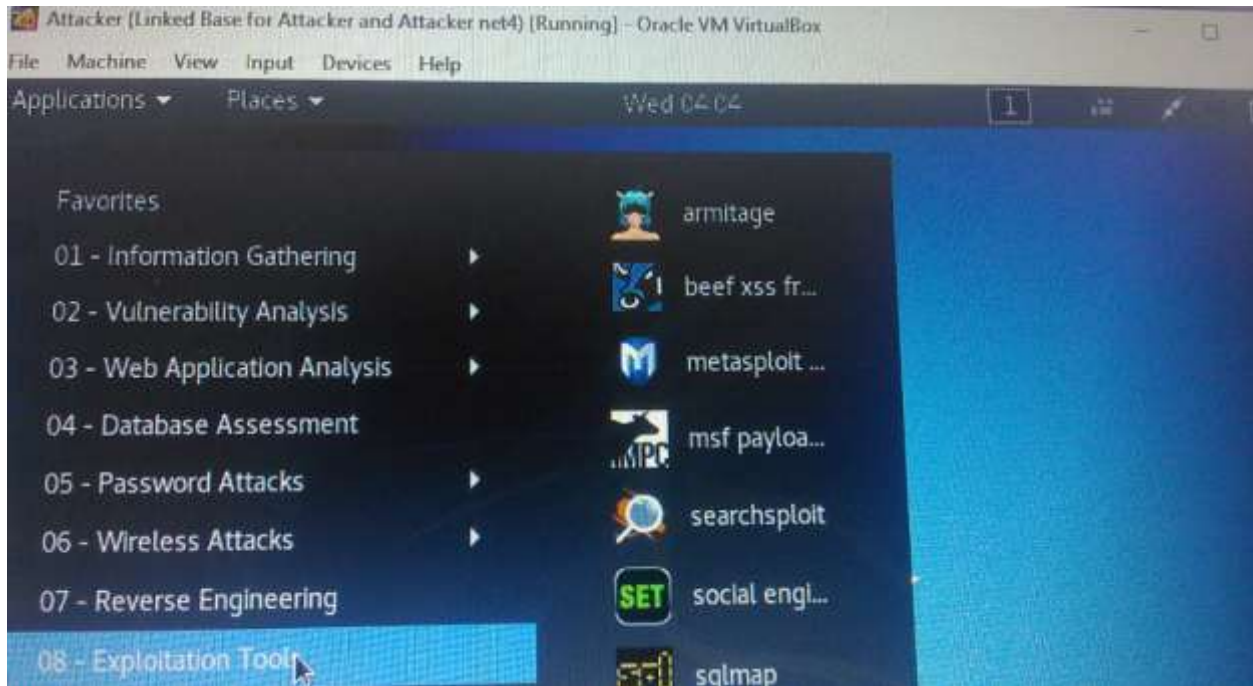


Figure 36: Exploiting Tools (Author, 2019)

Figure 36 shows different exploit tools that can be used to hack into PC's virtual machines. It shows that attackers have different tools. Figure 37 shows the exploit tool used by the researcher. Metasploit was used to hacking into the user's virtual machine.

4.4.6.4 Metasploit

Other servers used Metasploit for testing. IDS was used to test false negative. In the firewall, it was used to test if an external attacker can gain access to the network through a virtual machine. In figure 37 shows the tools used for scanning the network.

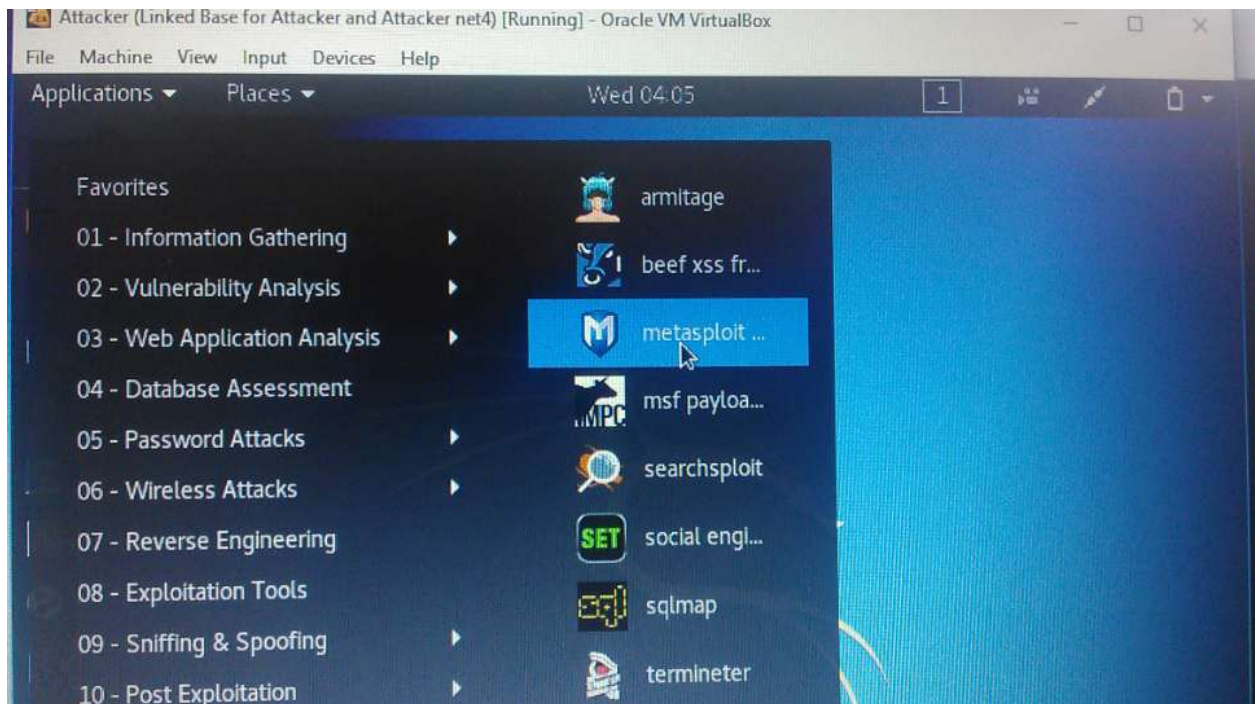


Figure 37: Metasploit (Author, 2019)

4.4.6.5 Launching an Exploit

Metasploit was used to launching an attack on the virtual machine. The external attacker to the virtual machine to gain access to the machine launches Metasploit. This is done through the network in our case through the IDS to demonstrate its weakness on the false negative.

Metasploit V5.0.10.dev was used as a hacking tool and file system was used to store the file. The exploit commands were used to launch the hacking tools. It was then found out that, the attacker gained access to the user's machine through the interpreter as mention in figure 38.

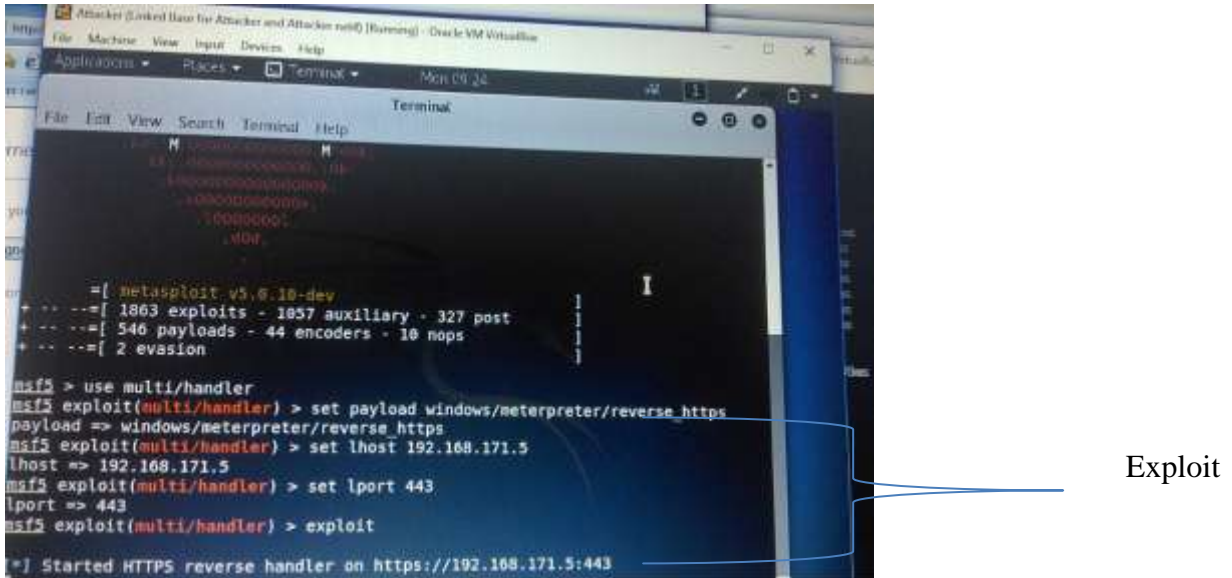


Figure 38: Launching an Exploit (Author, 2019)

4.4.6.6 Downloaded File

This is to show that the exploit has been sent and the virtual machine is requested to download the file.

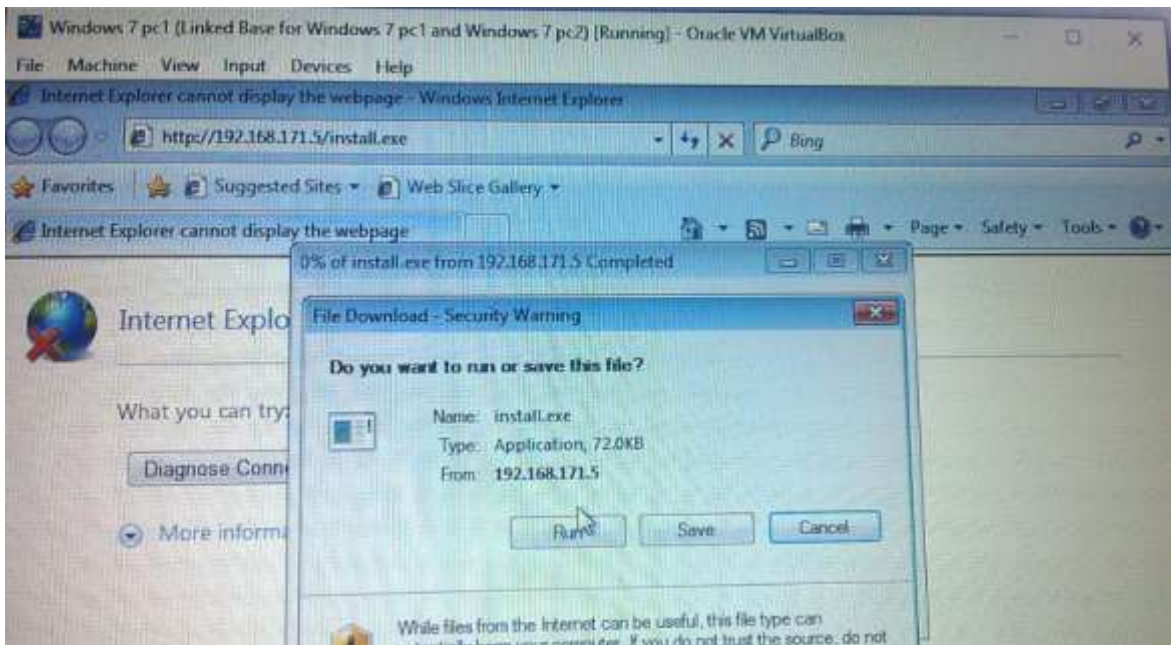


Figure 39: Downloaded (Author, 2019)

Figure 39 shows a virtual machine trying to run a file that contains an exploit. This is an example of a file that can be downloaded while it has been sent by an attacker. When the

virtual machine goes ahead to download the file then the attacker gains access to the machine. The information of the file is also indicated. That means the file is from the IP address of 192.168.171.5 and the URL is http://192/168.171.5/install.exe. Once the file is downloaded and saved the machine is automatically in the hands of the hacker who has an IP address of 192.168.171.5 as shown the figure 39.

4.4.6.7 Virtual machine versus the Attacker's machine

This shows proof that the attacker is accessing the virtual machine and can screenshot what the user in the virtual machine is doing in the machine. It also shows how the attacker saves the file and where it is saved.

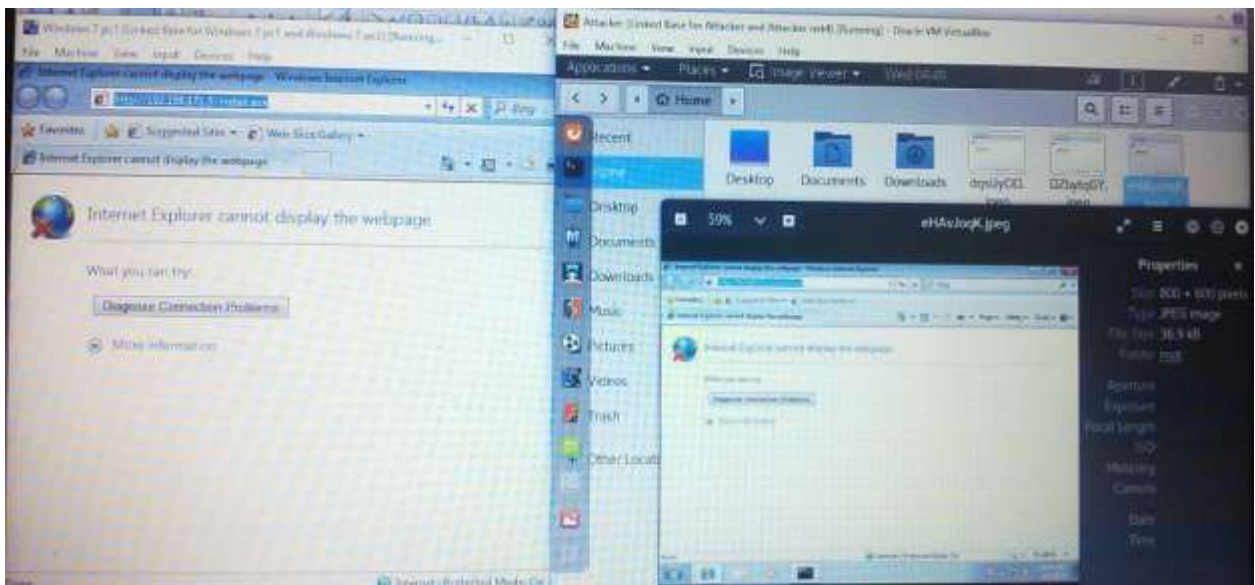


Figure 40: Virtual Machine versus the attackers' Machine (Author, 2019)

Figure 40 shows an attacked machine on the left and the right side shows that an attacker can see everything being done in the virtual machine. The attacker has gained access and can do anything with the virtual machine. Figure 40 right shows files downloaded by an attacker from the virtual machines where they are stored. All the information acquired in the virtual machine is stored in the home directory and can be accessed by the attacker.

4.4.7 False Negative

False negative occurs when the IDS fails to detect malicious activity. Although the issue of the IDS not being able to detect malicious data is still a problem and is something that still needs further investigation, reducing the false alarms should be the main priority.

4.4.7.1 IDS Router Connection

The IDS could not detect the attack on the virtual machine, which is on the network. The IDS connection is established with the virtual machine and the attacker but could not detect an attack thus demonstrating the false negative. The IDS in this research used the snort. This result is shown in figure 41

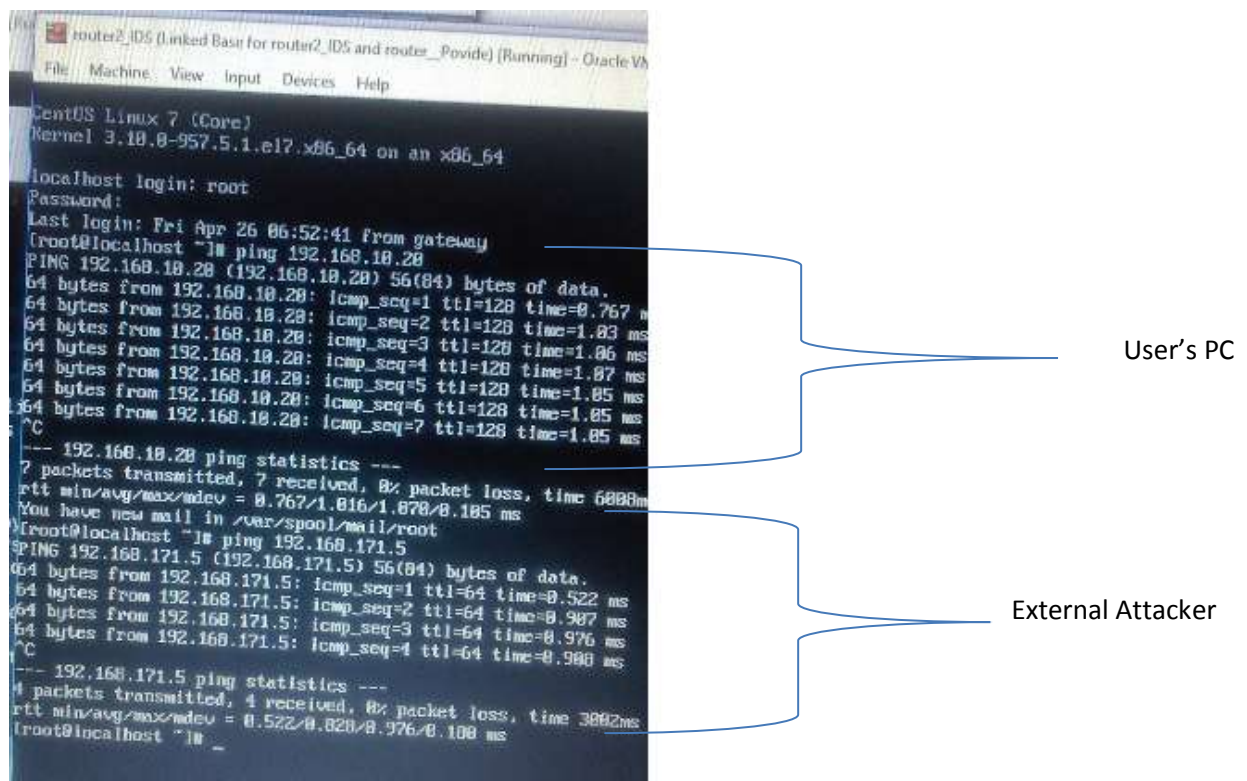


Figure 41: IDS Router Connection (Author, 2019)

In the IDS server, it shows that there is a communication between the snort with the external attacker and users' pc but it could not detect the attack used to attack the user's machine. It was found out that it could not report on the Metasploit file downloaded by the user as a hacking tool thus demonstrating the false negative. The IDS could connect to the

virtual machine with an IP address of 192.168.10.28 that is presumed to be under attack by the external attacker. It can also connect to the external attacker through the IP address of 192.168.171.5 without detecting an exploit. The results as shown in the figure above show communication between the server and the machine, and could not detect the attacks. The results were compared to that of Linora and Barathy (2014) who proposed an intrusion detection system that depends on the honey pot. They built the models of normal behavior for multitier web applications considering both front-end requests and backend database queries. The result of their work shows their approach is feasible and effective in reducing both false positives and false negatives. The IDS did not indicate the mechanism it would employ for detecting whether it is malicious activities or not. This can be curbed by analyzing the network and only sending an alert when the attack occurs. POVIDE Model would only send an alert upon detection of the model.

4.5 Design of POVIDE Model

The purpose of this section is to do a quick design and prototype building to realize a POVIDE Model that allows the processes to be achieved effectively in the cloud environment without necessarily modifying the functionality and/or infrastructure of the existing cloud architecture. POVIDE Model was developed from the perspective of its intended users.

4.5.1 Flowchart Diagram of POVIDE Model

In figure 42, it shows activities realized by the flowchart. It starts with a sign up to use the POVIDE Model, and then the Network Admin verifies the user's credentials. If the user is not valid, it takes you back to sign up page. If the user is valid then the user enters the username and password. If the username is ok the device is initialized else the user is taken back to sign up page. After that, the user connects to the network; POVIDE Model scans the network for the possibility of an open network. If there are any open ports it blocks the port

else the session is created on the cloud. VM1 accesses the site on the internet, if the threats are detected then the site is blocked (false positive). The external attacker sends an exploit to VM1 in order to attack. The model detects the threats and blocks the IP address where the threat has originated from, (false negative). If all the threats and perceived threats detected the session is created on the cloud.

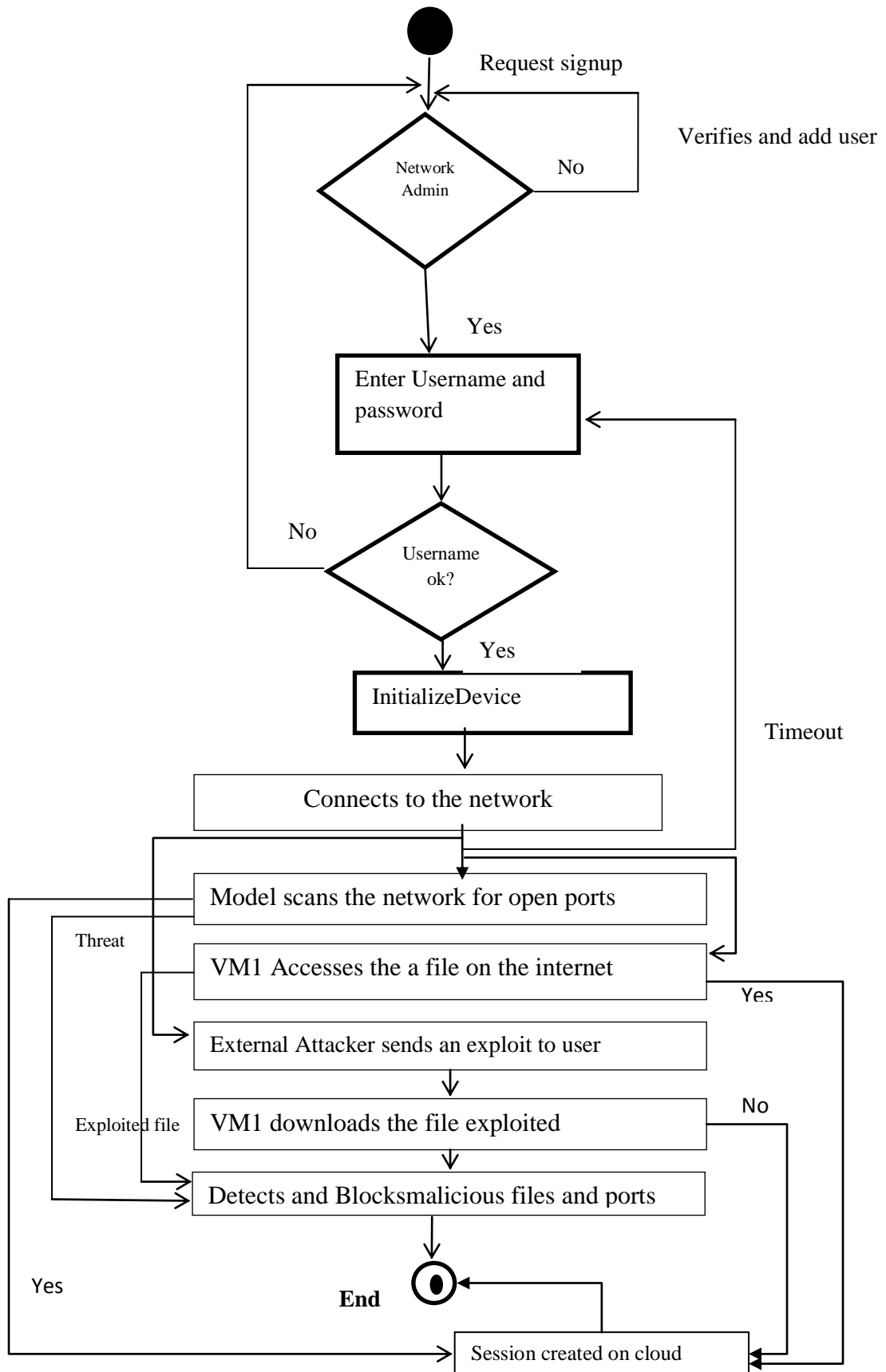


Figure 42: Flowchart Diagram of POVIDE Model (Author, 2019)

4.5.2 Policy Violation Detection Model Architecture

The user can access the POVIDE Model using different technologies such as laptops and phones. The user can access different cloud services depending on cloud service providers. The cloud is divided into layers. The upper layer is Software as a Service (SaaS), which is the one visible to the final user and involves applications. The next layer is Platform as a Service (PaaS) and it matters to software developers. It is composed of the operating systems, application programming interfaces (API), documentation, and basic services. Infrastructure as a Service (IaaS) refers to the usage of available resources on the cloud: memory, processors, storage and finally business process as a service (BPaaS) is the delivery of business process outsourcing (BPO) services that are sourced from the cloud and constructed for multitenancy. As a cloud service, the BPaaS model is accessed via Internet-based technologies. A cloud management platform is a suite of integrated software tools that an enterprise can use to monitor and control cloud computing resources. While an organization can use a cloud, management platform exclusively for private or public cloud deployment, these tools set target hybrid and multi-cloud models to help centralize control of various cloud-based infrastructures. Then there is the Policy Violation Detection Model that is used to detect any violation on the cloud see figure 43. This is a layer added to bridge the gap between the cloud and cloud services POVIDE model. The model scans the ports for the network and blocks the open ports. It also detects and blocks malicious applications before the session is created in the cloud.

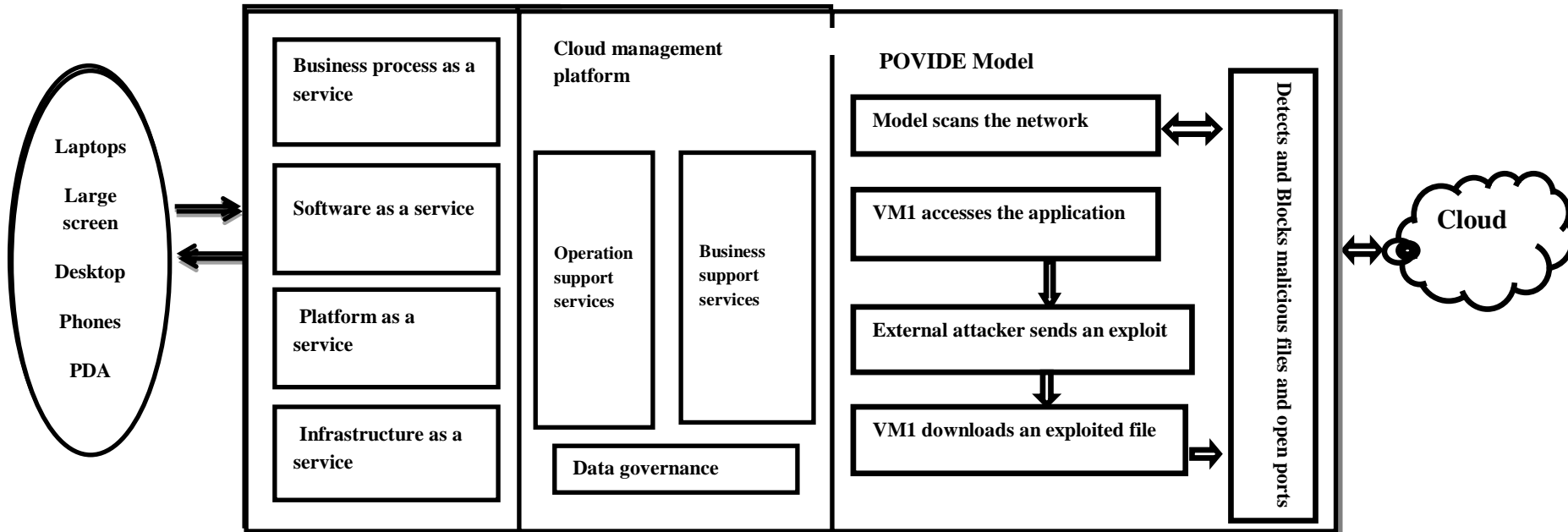


Figure 43: Policy Violation Detection Model Architecture (Author, 2019)

4.5.3 POVIDE Model Algorithm

Step 1: Start

Step 2 User: User request to be signed up in the POVIDE Model

Step 3 Actors: Verifies and add the user to the Model

Step 4: POVIDE Model allows the user to enter the username and password

- i. The user enters username and password
- ii. Model checks username

If username does not exist in the file system it sends invalid username to go to step 2

Step 5 Short descriptions: Login verifies user access to POVIDE Model based on username and password pair that is entered by the user on the end-user device (personal computers or iPad or smartphone)

Step 6 User initializes the devices used in the server at the same time

Step 7 Pre-condition: User is connected to the network (tested through IP address ping with VM1, VM2, internal attacker external attacker and the POVIDE Model)

Step 8 Invariant: If time out occurs, restarts the procedures go to step 3 above

Step 9: POVIDE Model scans the network for open ports and blocks the open ports

Step 10: VM1 accesses the website on the internet

Step 11: POVIDE Model checks the site. If it a threat block and sends a report, else the site is genuine go to step 16 (false positive)

Step 12: External Attacker sends an exploit file to VM1

Step 13: VM1 downloads the file from an external attacker

Step 14: POVIDE Model detects the threat and blocks the IP address. Else go to step 16 (false negative)

Step 15: Connection established in cloud

Step 16: End

4.5.4 POVIDE Model Experimental Setup

The experiment on the POVIDE model helps in curbing the weaknesses of firewalls and IDS as mentioned in the above demonstration. The experiment was done in two stages the scanning stage and the penetration stage. The aim of this third set of experiments was to test the following; Firstly, it was to test the false positive and false negative. Secondly, it provided insight into which tools are suitable for scanning the network for opened ports and

blocks. This latter point is the third objective of this research and the subject of the next chapter. This section describes the set of experiments that were undertaken in order to curb the weaknesses that were demonstrated using a firewall and IDS. Here 80GB of HDD and 3GB memory were used. The experiment was used to verify that the POVIDE model can solve the issue of false positive and false negative. The experiment further demonstrated that an internal attacker on the network is not able to scan the network. The model is made of four virtual machines. These are the machines, internal attacker and external attacker as shown in figure 44.

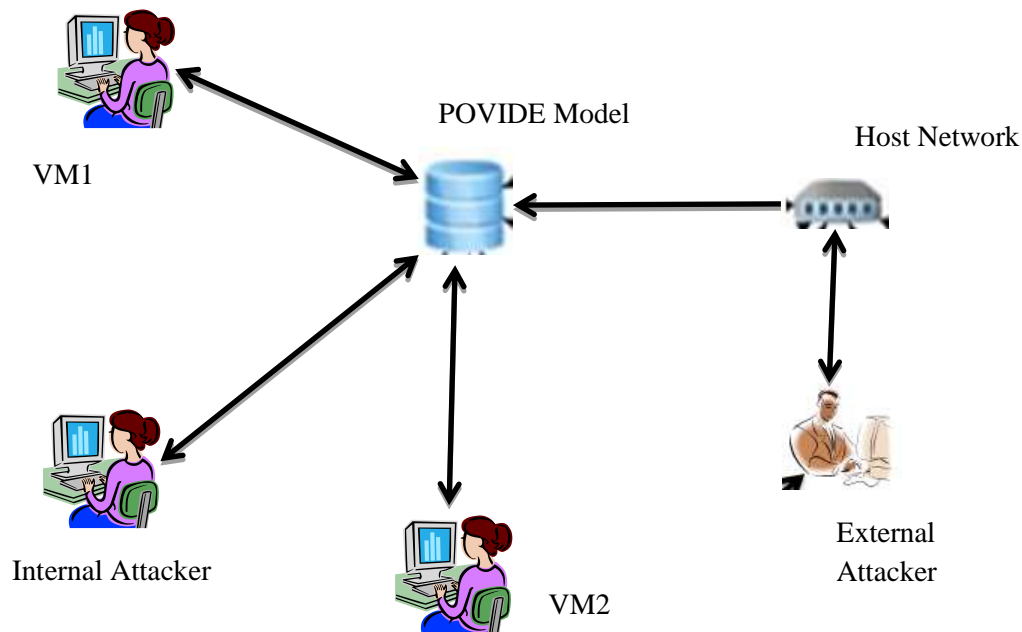


Figure 44: POVIDE Model Setup (Author, 2019)

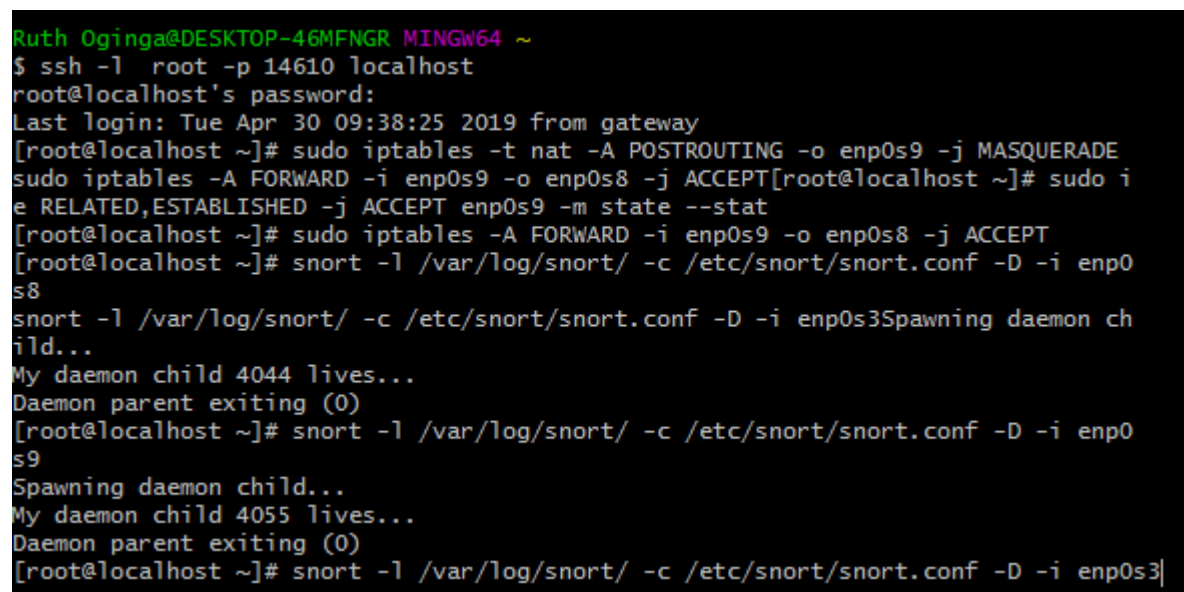
POVIDE Model scans the network for the open ports and blocks help curb threats of hackers. It also solves the issue of false positive and false negative. The model will not send an alert report when the user is access normal download that is not a threat. Lastly, the model detects the threat and blocks the IP address where the threat originates.

4.5.5 Configuration Stage

The `ssh -l root -p 14610 localhost` command was then used to securely operate network services over an unsecured network. Next `root@localhost`'s password was used to display the metadata on any physical volume on the system, as shown in figure 4.30. This was to allow only valid users to access the system. The configuration of the Ip tables was done by `sudo iptables -t nat -A POSTROUTING -o enp0s9 -j MASQUERADE` `sudo iptables -A FORWARD -I enp0s9 -o enp0s8 -j ACCEPT`, to enable packet forwarding by the kernel. Network-address-translation (NAT) on a Linux system with iptables rules so that the system can act as a gateway and provide internet access to multiple hosts on a local network using a single public IP address and the codes are shown in appendix V.3.

```
snort -l /var/log/snort/ -c /etc/snort/snort.conf -D -i enp0s8 ,snort -l /var/log/snort/ -c /etc/snort/snort.conf -D -i enp0s9 ,snort -l /var/log/snort/ -c /etc/snort/snort.conf -D -i enp0s3
```

The three snort commands were configured to log traffic from the three interfaces used. These are the user's machine, an external attacker and the model. The results of the traffic were shown in figure 45 and the code is placed in appendix V.1.



```
Ruth Oginga@DESKTOP-46MFNGR MINGW64 ~
$ ssh -l root -p 14610 localhost
root@localhost's password:
Last login: Tue Apr 30 09:38:25 2019 from gateway
[root@localhost ~]# sudo iptables -t nat -A POSTROUTING -o enp0s9 -j MASQUERADE
sudo iptables -A FORWARD -i enp0s9 -o enp0s8 -j ACCEPT[root@localhost ~]# sudo i
e RELATED,ESTABLISHED -j ACCEPT enp0s9 -m state --stat
[root@localhost ~]# sudo iptables -A FORWARD -i enp0s9 -o enp0s8 -j ACCEPT
[root@localhost ~]# snort -l /var/log/snort/ -c /etc/snort/snort.conf -D -i enp0
s8
snort -l /var/log/snort/ -c /etc/snort/snort.conf -D -i enp0s3Spawning daemon ch
ild...
My daemon child 4044 lives...
Daemon parent exiting (0)
[root@localhost ~]# snort -l /var/log/snort/ -c /etc/snort/snort.conf -D -i enp0
s9
Spawning daemon child..
My daemon child 4055 lives...
Daemon parent exiting (0)
[root@localhost ~]# snort -l /var/log/snort/ -c /etc/snort/snort.conf -D -i enp0s3|
```

Figure 45: Configuration Stage (Author, 2019)

4.5.6 Scanning Stage

The purpose of scanning is to see which port is open. The result of the scanning process was done on the POVIDE Model. The Model has the IP Address 192.168.60.60 which is the network server, which is made up of IDS and firewall server application. The Model is made up of two virtual machines with IP addresses of 192.168.60.80 and 192.168.60.91. One has opened port while the other the all the ports are closed. The computer with the IP address 192.168.60.80 is the address with closed ports while the IP address of 192.168.60 91 has opened ports. Open ports include port 2105, 445, 139, 49156,20107,49157,1881,5357,135, and 2103. Open ports are used for exploits.

4.5.6.1 Scanning

The POVIDE Model helps in scanning. This helps in such a way that an internal and external attacker cannot scan the network for the open ports. POVIDE model scans the entire network and blocks the ports that are opened.

After successfully logging in as root then scan the network through scan-results command. Scann-results command enables the POVIDE model to provide results. ./port-task command was used to scan the entire network for any opened port. Only one open port was found which a virtual machine with an IP address of 192.168.60.91. It also provides the list of ports opened as shown in the figure above the opened ports provided includes port 2105, 445,139, 49156,20107,49157,1881,5357,135, and 2103.cat blocked.txt command was used to view the content of the network of POVIDE model, showing the virtual machines that have been blocked. The report provided is as follows blocked IP 192.168.60.91 and access denied by the firewall. The results were as expected, as shown in Figure 46. It was found out that only one PC's port was opened and was blocked. This would enable to Model to be secure in such a way that an internal attacker or the external attacker cannot use the open port to hack

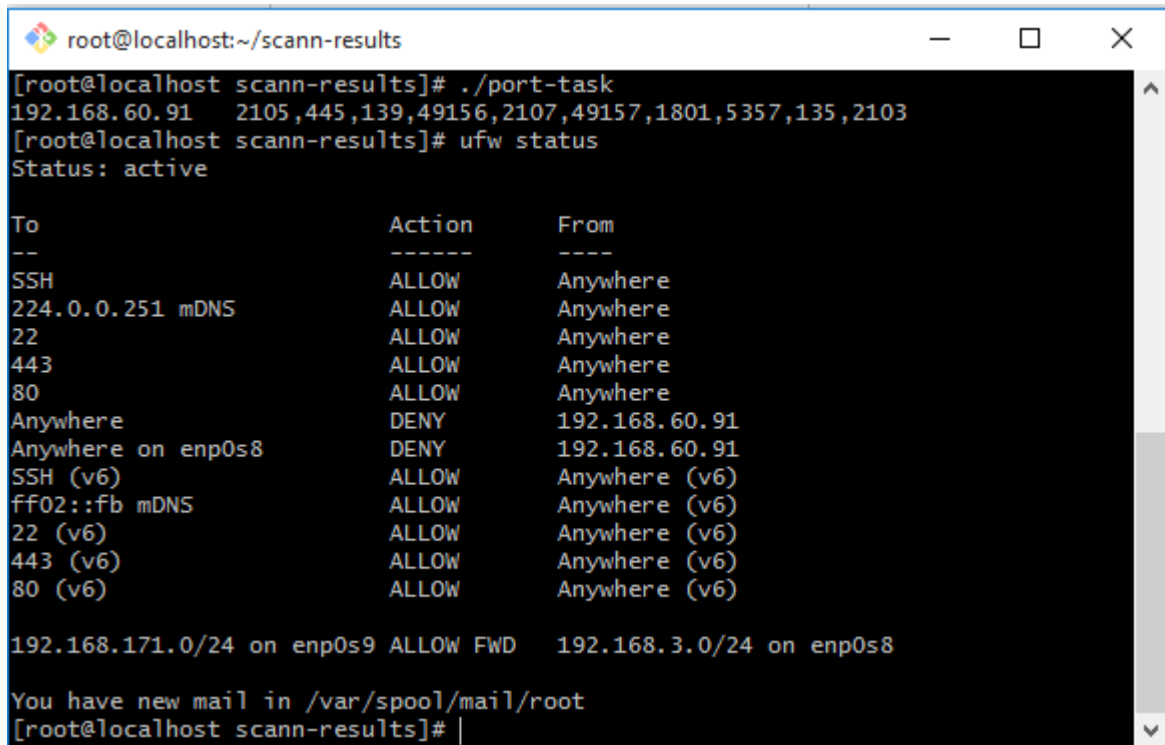
into the machine. This means the PROVIDE Model would block the internal and external attacks. This solves the weakness of the firewall and IDS weakness in two ways. First, it would block the internal attacker from scanning the network and it will not allow the external attacker to attack using the opened ports. Second, the Model will not require the network administrator to block the virtual machine as it would do it real time as long as the cat blocked.txt is configured on the model and the codes are shown in appendix V.5

```
root@localhost:~/scann-results  
Ruth Oginga@DESKTOP-46MFNGR MINGW64 /  
$ ssh -l root -p 14610 localhost  
root@localhost's password:  
Last login: Mon Jul 1 17:17:31 2019 from gateway  
[root@localhost ~]# cd scann_results  
-bash: cd: scann_results: No such file or directory  
[root@localhost ~]# cd scann-results  
[root@localhost scann-results]# ./task  
Skipping adding existing rule  
Firewall stopped and disabled on system startup  
Command may disrupt existing ssh connections. Proceed with operation (y|n)? y  
Firewall is active and enabled on system startup  
[root@localhost scann-results]# cat blocked.txt  
Found open ports in 192.168.60.91 and access blocked by ufw firewall  
Found open ports in 192.168.60.91 and access blocked by ufw firewall  
Found open ports in 192.168.60.91 and access blocked by ufw firewall  
Found open ports in 192.168.60.91 and access blocked by ufw firewall  
Found open ports in 192.168.60.91 and access blocked by ufw firewall  
Found open ports in 192.168.60.91 and access blocked by ufw firewall  
Found open ports in 192.168.60.91 and access blocked by ufw firewall  
Blocked IP: 192.168.60.91 and access denied by firewall  
Blocked IP: 192.168.60.91 and access denied by firewall  
Blocked IP: 192.168.60.91 and access denied by firewall  
Blocked IP: 192.168.60.91 and access denied by firewall  
Blocked IP: 192.168.60.91 and access denied by firewall  
[root@localhost scann-results]# ./port-task  
192.168.60.91 2105,445,139,49156,2107,49157,1801,5357,135,2103  
[root@localhost scann-results]# |
```

Figure 46: Scanning (Author, 2019)

4.5.6.2 Allowed and Denied Ports

The command groups the ports into two. It either denies or allows the port and IP addresses. The ports that are allowed include 443 and 88 respectively. The denied IP address is 192.168.60.91 and it is denied on any port.

A terminal window titled 'root@localhost:~/scann-results' showing the output of the 'ufw status' command. The output lists various ports and their actions, including 'ALLOW' for ports 22, 443, 80, and 'DENY' for port 443 from IP 192.168.60.91. It also shows firewall rules for network ranges 192.168.171.0/24 and 192.168.3.0/24.

```
root@localhost:~/scann-results
[root@localhost scann-results]# ./port-task
192.168.60.91 2105,445,139,49156,2107,49157,1801,5357,135,2103
[root@localhost scann-results]# ufw status
Status: active

To Action From
--
SSH ALLOW Anywhere
224.0.0.251 mDNS ALLOW Anywhere
22 ALLOW Anywhere
443 ALLOW Anywhere
80 ALLOW Anywhere
Anywhere DENY 192.168.60.91
Anywhere on enp0s8 DENY 192.168.60.91
SSH (v6) ALLOW Anywhere (v6)
ff02::fb mDNS ALLOW Anywhere (v6)
22 (v6) ALLOW Anywhere (v6)
443 (v6) ALLOW Anywhere (v6)
80 (v6) ALLOW Anywhere (v6)

192.168.171.0/24 on enp0s9 ALLOW FWD 192.168.3.0/24 on enp0s8

You have new mail in /var/spool/mail/root
[root@localhost scann-results]# |
```

Figure 47: Allowed and Denied Ports (Author, 2019)

Then the **ufw status** command was used to display the machines that are active and the ones that have been blocked. It was found out that only one was not active because it was blocked for having an open port on the network. When a port is opened scanning and exploitation become very easy. The results also show the machines that are active that means all the ports are closed. These are machines on 192.168.171.0/24 network and 192.168.3.0/24 as shown in figure 47 and the codes are available in appendix V.5.

4.5.7 Penetration Stage

Penetration is done by sending all the exploits from the attacker's computer to the server computer through open ports. Open ports are obtained from the scanning stage. The delivery exploits have the objective to do penetration. Penetration was conducted in order to test the model for false negative or false positive.

4.5.7.1 False Positive

As shown in figure 34 the IDS server that uses snort detects the website as an alert. It detects the web site as a live attack. When it comes to the PROVIDE model the user machine accesses the same website but it will not report it as an attack but a normal website. The results were as expected as shown in figure 48.

4.5.7.2 Normal Site

This shows a normal site that is not exploited by the attacker. This site of Apache2 Debian was used and was found in <http://192.168.171.5/> as shown in figure 48.

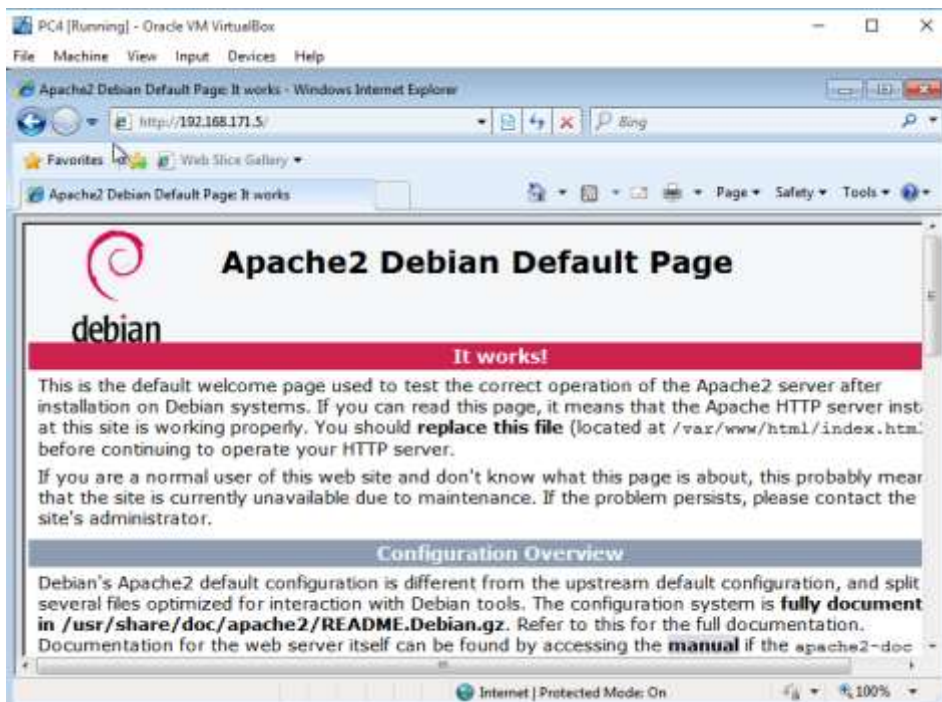


Figure 48: Normal Site (Author, 2019)

4.5.7.3 Results of Accessing Normal Site

The model reports the website access in figure 48 as normal and the IP address accessing the website as shown in figure 49.

```
root@localhost:~#
9282
07/01-18:15:50.259760 [**] [1:10000004:0] HTTP Packet found [**] [Priority: 0] {TCP} 192.168.171.5:80 -> 192.168.171.8:49281
07/01-18:15:50.260522 [**] [1:10000004:0] HTTP Packet found [**] [Priority: 0] {TCP} 192.168.171.8:49281 -> 192.168.171.5:80
07/01-18:15:53.410395 [**] [1:10000004:0] HTTP Packet found [**] [Priority: 0] {TCP} 192.168.171.8:49281 -> 192.168.171.5:80
07/01-18:15:53.410809 [**] [1:10000004:0] HTTP Packet found [**] [Priority: 0] {TCP} 192.168.171.5:80 -> 192.168.171.8:49281
07/01-18:15:53.411669 [**] [1:10000004:0] HTTP Packet found [**] [Priority: 0] {TCP} 192.168.171.8:49283 -> 192.168.171.5:80
07/01-18:15:53.412069 [**] [1:10000004:0] HTTP Packet found [**] [Priority: 0] {TCP} 192.168.171.5:80 -> 192.168.171.8:49283
07/01-18:15:53.412518 [**] [1:10000004:0] HTTP Packet found [**] [Priority: 0] {TCP} 192.168.171.8:49283 -> 192.168.171.5:80
07/01-18:15:53.412583 [**] [1:10000004:0] HTTP Packet found [**] [Priority: 0] {TCP} 192.168.171.8:49283 -> 192.168.171.5:80
07/01-18:15:53.412726 [**] [1:10000004:0] HTTP Packet found [**] [Priority: 0] {TCP} 192.168.171.5:80 -> 192.168.171.8:49283
07/01-18:15:53.414741 [**] [1:10000004:0] HTTP Packet found [**] [Priority: 0] {TCP} 192.168.171.8:49283 -> 192.168.171.5:80
07/01-18:15:53.417564 [**] [1:10000004:0] HTTP Packet found [**] [Priority: 0] {TCP} 192.168.171.8:49283 -> 192.168.171.5:80
07/01-18:15:53.418162 [**] [1:10000004:0] HTTP Packet found [**] [Priority: 0] {TCP} 192.168.171.5:80 -> 192.168.171.8:49283
07/01-18:15:53.625628 [**] [1:10000004:0] HTTP Packet found [**] [Priority: 0] {TCP} 192.168.171.5:80 -> 192.168.171.8:49283
07/01-18:15:53.626117 [**] [1:10000004:0] HTTP Packet found [**] [Priority: 0] {TCP} 192.168.171.8:49283 -> 192.168.171.5:80
07/01-18:15:55.321850 [**] [1:10000004:0] HTTP Packet found [**] [Priority: 0] {TCP} 192.168.171.8:49283 -> 192.168.171.5:80
07/01-18:15:55.323833 [**] [1:10000004:0] HTTP Packet found [**] [Priority: 0] {TCP} 192.168.171.8:49283 -> 192.168.171.5:80
07/01-18:15:55.327410 [**] [1:10000004:0] HTTP Packet found [**] [Priority: 0] {TCP} 192.168.171.8:49283 -> 192.168.171.5:80
07/01-18:15:55.328096 [**] [1:10000004:0] HTTP Packet found [**] [Priority: 0] {TCP} 192.168.171.5:80 -> 192.168.171.8:49283
07/01-18:15:55.534025 [**] [1:10000004:0] HTTP Packet found [**] [Priority: 0] {TCP} 192.168.171.5:80 -> 192.168.171.8:49283
07/01-18:15:55.534964 [**] [1:10000004:0] HTTP Packet found [**] [Priority: 0] {TCP} 192.168.171.8:49283 -> 192.168.171.5:80
```

Figure 49: False positive (Author, 2019)

Vi /etc/snort/rules/local.rules command was used to display the results in POVIDE Model. The results show the packets found, the IP address accessing the site and the IP address where the site is being accessed. The POVIDE Model does not send false alerts from the server thus solve the issues of false positive in most IDS. The model would only send a log file, which contains the web site being accessed; the web site is an HTML file. This test can prove that POVIDE Model would eliminate false positives.

4.5.7.4 False Negative

As seen in figure 4.26 the IDS server using snort could not detect the attack that happened. When using the POVIDE model, the attack was detected and blocked the attack. Figure 50 shows an example of a threat sent by an attacker and accessed by the virtual machine. Figure 50 is the POVIDE Model shows the results after detecting and blocking the attack on the VM. It shows the name of the threat, where it's coming from and which IP address affected. It shows that the POVIDE Model would detect and block the attacks in real

time thus, eliminating the false negative. It monitors the events on the network, inspects the data and collects evidence of intrusive behaviors. Whenever it detects suspicious or malicious attempts, it signals an administrator instantly for a reaction.

4.5.7.5 Exploited Application

This is an example of an application that has been exploited with the attacker. The file use was found in <http://192.168.171.5/install.exe> as shown in figure 4.35.

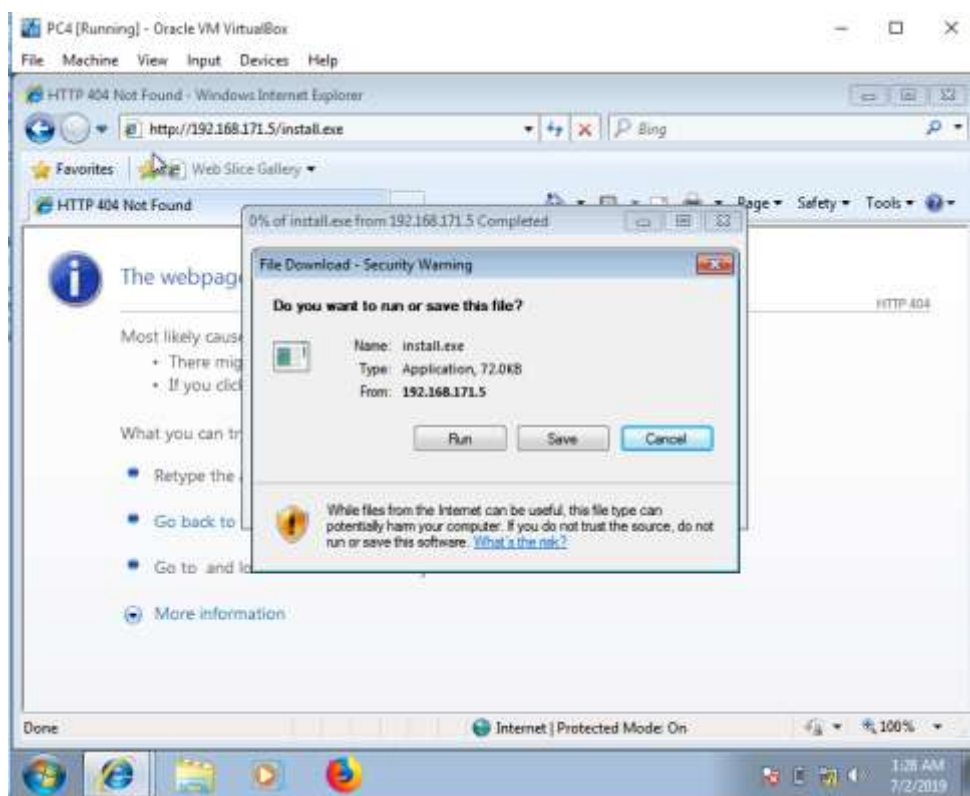


Figure 50: Exploited Application (Author, 2019)

4.5.7.6 Threat Detected and Blocked

This shows that the exploited file in figure 51 has been detected and blocked. The file has a malicious code of Trojan and was exploited using Metasploit as shown in figure 51. This is the result of the external attacker trying to attack a user's Virtual machine on the PROVIDE model. Meterpreter is the exploiting tool. The file being accessed has been

forwarded to the POVIDE model using root/dqsiJyoD.jpeg for analysis. The idle time command is to show the time the machine has been idle. Shell is a command to show the kind of machine that is being attacked. The results are as shown in figure 51. The POVIDE Model is able to detect an attack as it happens if the attacker bay passes the blocked ports. The model would also check on the different processes created. This would help in detecting the false negative. This experiment showed that any action performed inside the VM is not recorded in the log file. Examples include restarting a VM, formatting a disk, and copying a file. On the other hand, actions performed via POVIDE Model or CLI are recorded the codes are shown in appendix V.4.

```

root@localhost:~
07/01-18:36:16.470237 [**] [1:10000005:0] Metasploit traffic detected and blocked [**] [Classification: A Network Trojan was Detected] [Priority: 443]
07/01-18:36:16.470759 [**] [1:10000005:0] Metasploit traffic detected and blocked [**] [Classification: A Network Trojan was Detected] [Priority: 443]
07/01-18:36:16.471363 [**] [1:10000005:0] Metasploit traffic detected and blocked [**] [Classification: A Network Trojan was Detected] [Priority: 9291]
07/01-18:36:16.472589 [**] [1:10000005:0] Metasploit traffic detected and blocked [**] [Classification: A Network Trojan was Detected] [Priority: 9291]
07/01-18:36:16.473868 [**] [1:10000005:0] Metasploit traffic detected and blocked [**] [Classification: A Network Trojan was Detected] [Priority: 443]
07/01-18:36:16.474450 [**] [1:10000005:0] Metasploit traffic detected and blocked [**] [Classification: A Network Trojan was Detected] [Priority: 9291]
07/01-18:36:26.475562 [**] [1:10000005:0] Metasploit traffic detected and blocked [**] [Classification: A Network Trojan was Detected] [Priority: 443]
07/01-18:36:26.476375 [**] [1:10000005:0] Metasploit traffic detected and blocked [**] [Classification: A Network Trojan was Detected] [Priority: 9292]
07/01-18:36:26.477117 [**] [1:10000005:0] Metasploit traffic detected and blocked [**] [Classification: A Network Trojan was Detected] [Priority: 443]
07/01-18:36:26.477676 [**] [1:10000005:0] Metasploit traffic detected and blocked [**] [Classification: A Network Trojan was Detected] [Priority: 443]
07/01-18:36:26.478105 [**] [1:10000005:0] Metasploit traffic detected and blocked [**] [Classification: A Network Trojan was Detected] [Priority: 9292]

```

Figure 51: False negative (Author, 2019)

4.5.8 Analysis of Logs

The logs sent for analysis are tcpdump.log.1556705453, snort.log.1556705351 as shown above. The result was the traffic shown above. `ls -lrt /var/log/snort/` this command shows all the directory of the encrypted files, including those with and without data. It shows the date and time of analyzing data and the size of data. The results of this are shown in the

figure above. `cat /var/log/snort/snort.log.1556705351` command was used to decrypt the file and log the traffic of the file. The results are shown in figure 4.37 and 4.38.

```

root@localhost:~
-rw-----, 1 root root 8324 Apr 25 10:02 tcpdump.log.1556198981
-rw-----, 1 root root 744 Apr 26 05:31 tcpdump.log.1556271100
-rw-----, 1 root root 384 Apr 26 05:38 tcpdump.log.1556271375
-rw-----, 1 root root 0 Apr 26 05:43 snort.log.1556271806
-rw-----, 1 root root 0 Apr 26 05:43 tcpdump.log.1556271831
-rw-----, 1 root root 0 Apr 26 05:43 snort.log.1556271831
-rw-----, 1 root root 0 Apr 26 05:44 tcpdump.log.1556271842
-rw-----, 1 root root 0 Apr 26 05:44 snort.log.1556271842
-rw-----, 1 root root 384 Apr 26 05:44 tcpdump.log.1556271822
-rw-----, 1 root root 680 Apr 26 05:44 snort.log.1556271822
-rw-----, 1 root root 0 Apr 26 06:09 tcpdump.log.1556273376
-rw-----, 1 root root 0 Apr 26 06:09 snort.log.1556273376
-rw-----, 1 root root 0 Apr 26 06:09 tcpdump.log.1556273380
-rw-----, 1 root root 0 Apr 26 06:09 snort.log.1556273380
-rw-----, 1 root root 34362 Apr 26 06:43 tcpdump.log.1556273375
-rw-----, 1 root root 46362 Apr 26 06:43 snort.log.1556273375
-rw-----, 1 root root 0 Apr 30 09:28 tcpdump.log.1556630890
-rw-----, 1 root root 0 Apr 30 09:28 snort.log.1556630890
-rw-----, 1 root root 170410 Apr 30 12:04 tcpdump.log.1556630529
-rw-----, 1 root root 229010 Apr 30 12:04 snort.log.1556630529
-rw-----, 1 root root 0 May 1 06:09 tcpdump.log.1556705352
-rw-----, 1 root root 0 May 1 06:09 snort.log.1556705352
-rw-----, 1 root root 0 May 1 06:09 tcpdump.log.1556705368
-rw-----, 1 root root 0 May 1 06:09 snort.log.1556705368
-rw-----, 1 root root 0 May 1 06:10 tcpdump.log.1556705453
-rw-----, 1 root root 37544 May 1 06:42 tcpdump.log.1556705351
-rw-----, 1 root root 50976 May 1 06:42 snort.log.1556705351
You have new mail in /var/spool/mail/root
[root@localhost ~]# cat /var/log/snort/snort.log.1556705351
HL\pH
Q:\v\pH\pH
QZ3' ^ '$: 4\pH
DCr\pH\pH
1V' EH9 DC4
kali7w
,y#HL\pI:v\pI\pI Z3' ^ '$: HL\pI

```

Figure 52: Analysis of logs (Author, 2019)

4.5.8.1 Analysis of Tcpdump.log

The results were analyzed and decrypted. It's upon the network admin to act on it by blocking the IP address that has sent it.

```

calhost ~]# cat /var/log/snort/tcpdump.log.1556630529
bL\
'cSc5WV' EH9 DC4 (I
kali7w
,/y*!L\R
W' EH9 DC4 (I' cSc5
kali7w
,/y*!L\I
DVV' EH9 DC4 r(I' cSc5
kali7w
,/y*!L\qZZ3' ^`$:'L\W
33' ^`$:'L\W
'@]v'L\W' EH9 DC4' cSc5
kali7w
,/y*!(L)pZZ3' ^`$:'
-L\,W' EH9 DC4' cSc5
kali7w
,/y*!5L\W' EH9 DC
4' cSc5

```

Figure 53: Analysis of tcpdup.log (Author, 2019)

4.5.8.2 Attack Responses Rules

The result in figure 54 indicates the response rules that the POVIDE model uses in the alert. It gives the date and time of the alert and the name of the attacking tools and the codes are shown in appendix V.2.

```

#-----
# ATTACK-RESPONSES RULES
#-----
#alert ip any any -> any any (msg:"ATTACK-RESPONSES id check returned root"; content:"uid=0|28|root|29|"; classtype:bad-unknown; sid:498; rev:6)
[root@localhost ~]# cat /etc/snort/rules/meterpreter.rules
#####
# abuse.ch URLhaus IDS ruleset (Snort / Suricata) #
# Last updated: 2019-03-26 05:53:04 (UTC) #
# #
# Terms Of Use: https://urlhaus.abuse.ch/api/ #
# For questions please contact urlhaus [at] abuse.ch #
#####
#
# url
#
#alert tcp $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS (msg:"Metasploit Meterpreter"; flow:to_server,established; content:"RCV"; http_client_body; depth:4; fast_pattern; isdataat:!0,relative; urilen:23<24,norm; content:"POST"; pcre:"/\^[a-z0-9]{4,5}_[a-z0-9]{16}\$/Ui"; classtype:trojan-activity; reference:url,blog.dierstevens.com/2015/05/11/detecting-network-traffic-from-metasploits-meterpreter-reverse-http-module/; sid:1618008; rev:1;)

```

Figure 54: Attack Responses Rules (Author, 2019)

4.6 Proof of Concept

A proof of concept is a closed but working solution, which can be evaluated and tested subject to clear criteria, from the understanding required to delivering success.

4.7 Model Quick Design and Prototype Building

The purpose of this section is to do a quick design and prototype building to realize a POVIDE Model that allows the processes to be achieved effectively in the cloud environment without necessarily modifying the functionality and/or infrastructure of the existing cloud architecture.

POVIDE Model was developed from the perspective of its intended users. The development process started with gathering the requirement from a survey of the literature. This was done through the Use Case Diagram. The Use Case Diagram was mapped to the requirements model to define exact model functionality. From the details of Use Case, the Sequence Diagram was constructed to describe the way different design components interact in the architecture. From Sequence Diagram a Class Diagram was created. This is a precise specification of services in the model. From the Class Diagram, a Component Diagram was built to define the logical Classes. Finally non-functional was incorporated into the model as shown in figure 55.

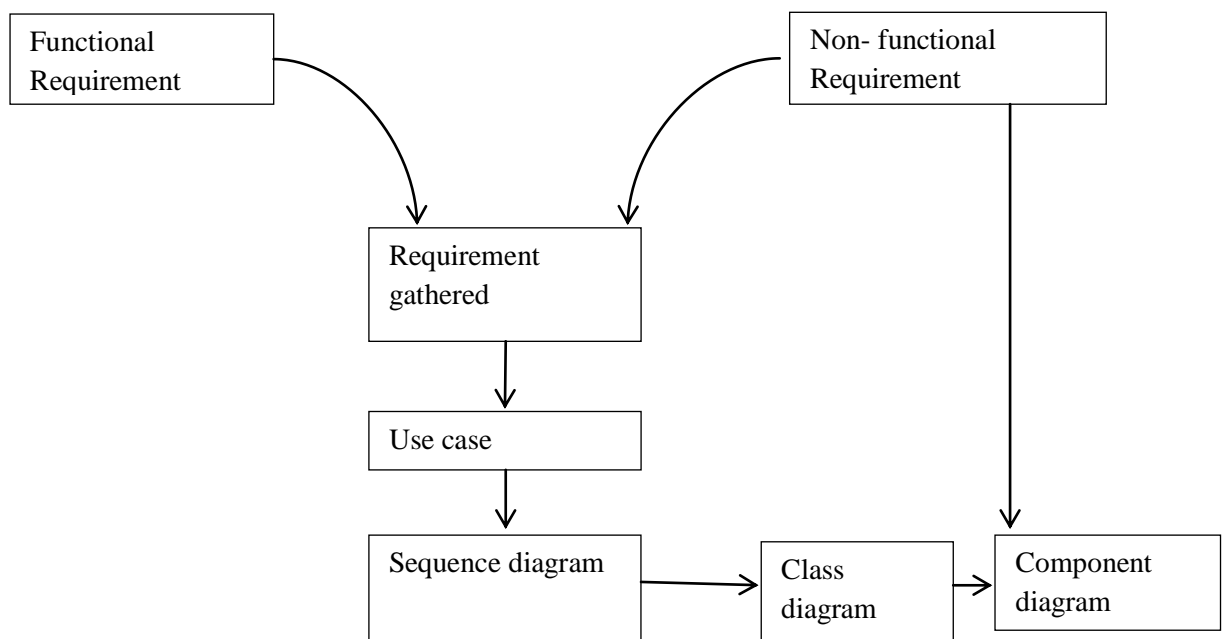


Figure 55: Model Quick Design (Author, 2019)

4.7.1 Requirement Gathering

As shown in figure 56 requirements are the collection of needs from the survey of literature while considering constraints under which the model must operate. The requirements of the POVIDE Model were informed by a review of the literature with a similar model. The survey was categorized into functional and non-functional requirements. Functional requirements are as follows policies; connectivity and access control while Non-functional requirement is not straight forward the requirement of the model rather it is related to usability and it includes performance, security, and elasticity.

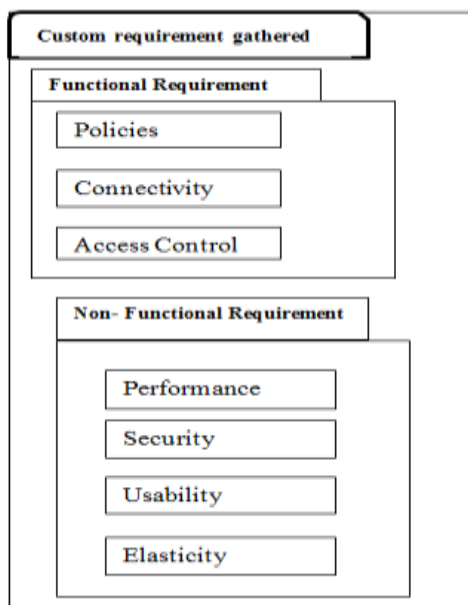


Figure 56: Requirement Gathering (Author, 2019)

4.7.1.1 Functional Requirement

POVIDE Model has three main functional requirements that need to be achieved these include Policies, connectivity and access control as shown in figure 57

Functional Requirement	
Connectivity	
	User's machine shall connect to either wireless or LAN
Access Control	
	POVIDE Model shall allow access to authorized user pc only
Policies	
	POVIDE Model shall check for false positive, false negative and network scanning by users' machines

Figure 57: Functional Requirement (Author, 2019)

4.7.1.2 Non-Functional Requirements

Non-functional requirements for POVIDE Model are the performance, security, usability, and elasticity as shown in figure 58.

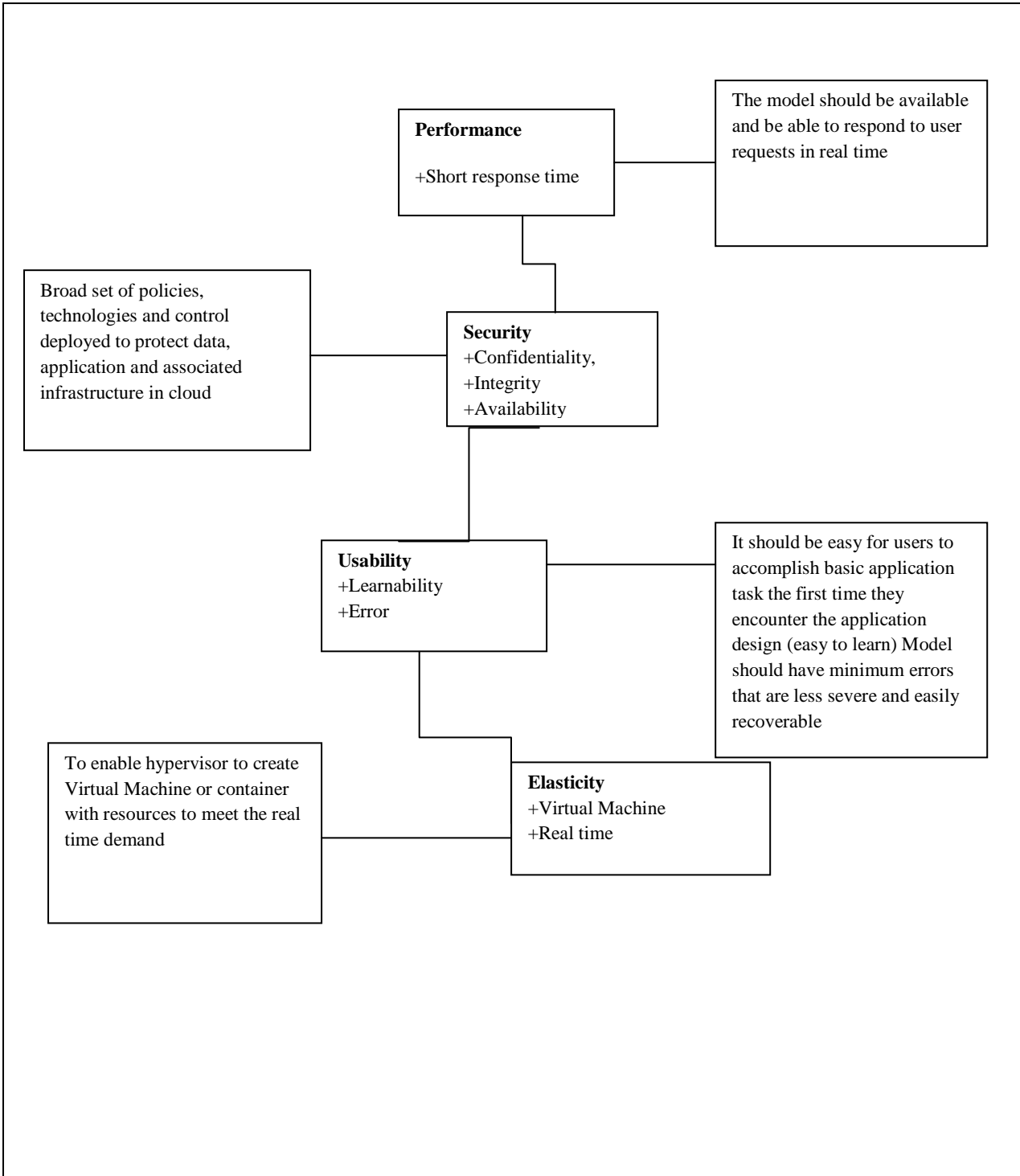


Figure 58: Non- Functional Requirements for POVIDE Model (Author, 2019)

4.7.2 Use Case Diagram for POVIDE Model

The Use Case model is a catalog of architecture functionality described using UML use case. The use case represents a single repeatable interaction that a user or actor experiences when using POVIDE Model. A use case typically includes one or more scenarios that describe interactions that go on between the actor and the Model, document the results and the larger pattern of interaction and may also be extended by other use cases to handle the conditions. A use case diagram captures use case and relationships between the user and the Model. It describes the functional requirements of the Model.

POVIDE Model consists of one use case that depicts the whole process. The actors are users, end-user devices: personal computers, mobile phones, and I pads. Access control authorizes and authenticates users'. Network Admin then verifies and add users to the cloud, File system stores passwords and data. Figure 59 shows the relationships of the POVIDE Model use case.

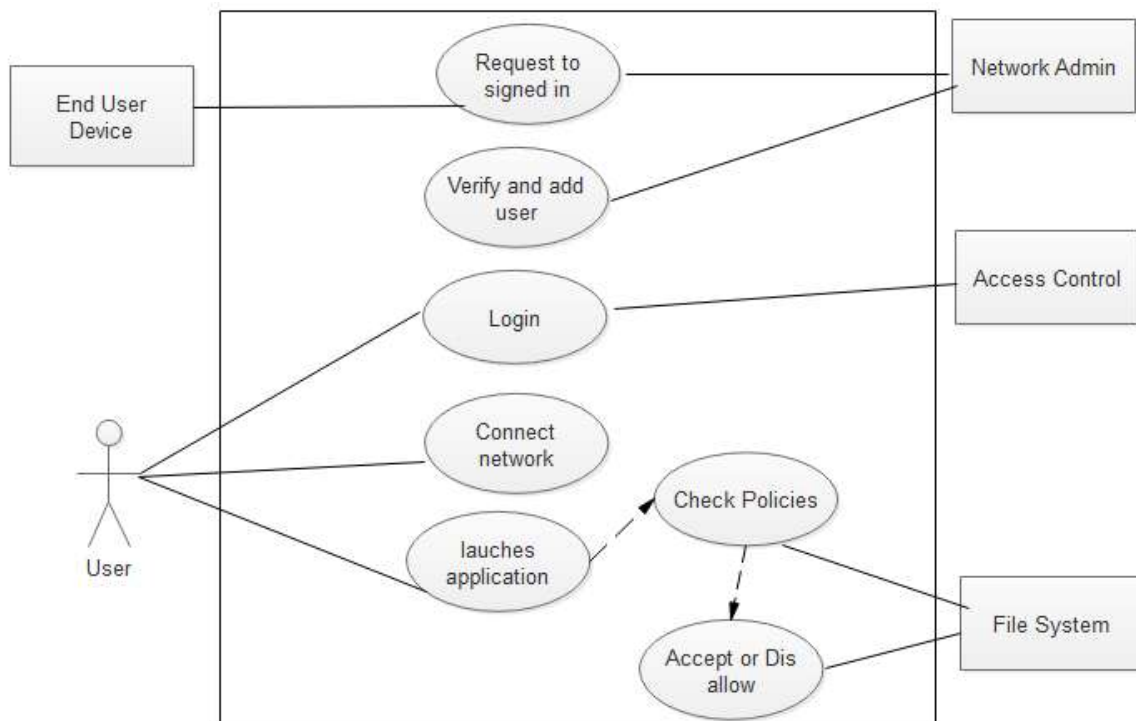


Figure 59: Use Case Diagram of POVIDE Model (Author, 2019)

4.7.3 Sequence Diagram for POVIDE Model

The login sequence diagram in figure 60 provides details of the object referenced in the interaction overview diagram. The sequence diagram is complete information about event flow between the user and the model.

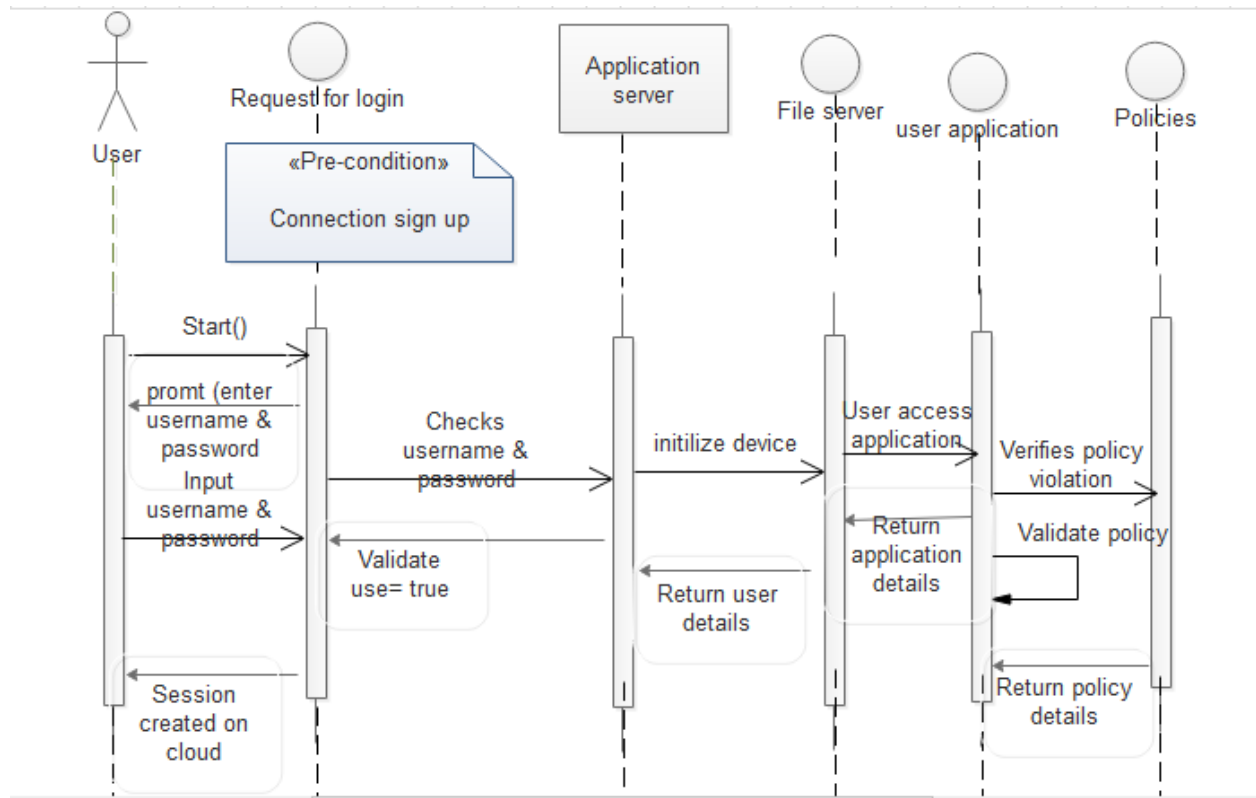


Figure 60: Sequence Diagram of POVIDE Model (Author, 2019)

The user starts by logging in to the Model, if not valid the application request for the signup form by the Network administrator. The user is allowed to enter the username and password. The request for login sends credentials to the application server, which in turn queries the user details. If the username and password exist in the file system then the user initializes the device. When the device is initialized, the user can access the application they want after that policy is verified in the application requested and send back a report by blocking or creating the session on cloud depending on the policy violation report.

Constrain in request to login sequence diagram define preconditions that should be satisfied before execution. Pre-condition is the existence of a connection between the user and the POVIDE Model. Connection signs up mean that the user must sign up in the model before being allowed to log in as shown in figure 60.

4.7.4 Class Diagram

The class diagram is a logical representation of the software system under construction. Classes generally have a direct relationship to source code or other software artifacts that can be grouped together into executable components. The model contains classes and artifacts which are being built or designed as part of the current model as well as classes and components that have been designed and built earlier and are being reused.

The class diagram is derived from the sequence diagram. Every communicating object in the sequence diagram represents a class while the message flows between classes form association. One sequence diagram represents the flow of the POVIDE Model as shown in figure 61. The sequence diagram has five communicating objects these are Request login, Server application, File server, user application, and Policies.

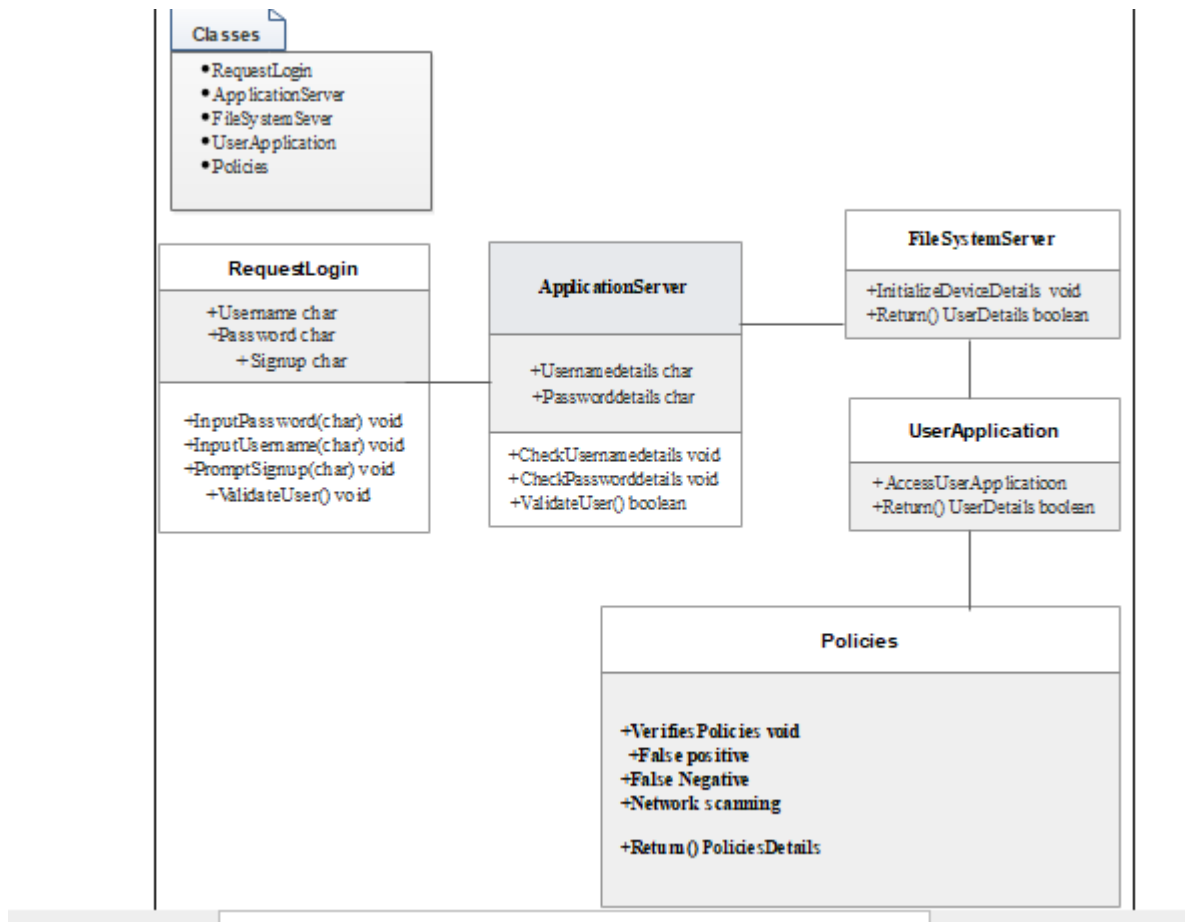


Figure 61: Class Diagram (Author, 2019)

The request logic class has password, Username, and signup as attributes and PromptsSignup (char), input user name (char), Inputpassword (char) and validateuser() as methods. This class provides a mechanism for controlling access to the model. The validateuser() method sends a request to the applicationserver to validate the user based on Username and Password. Applicationserver class, in turn, invokevaliduser() method which checks for the existence of username and Password in Filesystemserver. This is an iterative process that is managed by Return () userdetails method. From the sequence diagram, it is clear that requestlogin::Signup attribute is set to the return value of the Applicationserver. Applicationserver class has the usernameDetails and passwordDetails that mean that the applicationserver sends a query to check or search for the details of the user if they exist. It

then validates the user by retrieving the valid details. Upon retrieving the userdetails() it forwards the details to request login in order for the user to proceed.

Filesystemserver class has methods such as initializedevicedetails and return()userdetails.it allows the user to use the device of choice that has a valid username and password to establish the connection to the cloud network.

UserApplication class has methods such as Accessaplicationcloud and return()userdetails. Here the user accesses the application of choice and it then returns () users details so that Network Admin knows whether the user has established a connection to the network cloud.

Policies class has Verifiespolicy, policy violation and return() policydetails. Policy class verifies policy depending on the application the user is accessing. It then checks whether the violation has been made or not. If the violation is made the model detects the threat and blocks the port and IP address of the machine that has launched the threat. If the violation is not made then the connection is established in the cloud see figure 61.

4.7.5 Component Diagram

The main component of the PROVIDE Model is login request, application server, user application, file system and policies. The file system is composed of the login details and blocked ports. Login request contains credentials of logging in to the model while the application server verifies the credentials of logging in details of the users. The user application component contains the applications, which the user is accessing. Policies component contains the criteria used and that are put in place for the user to access the applications to curb violation as shown in figure 62.

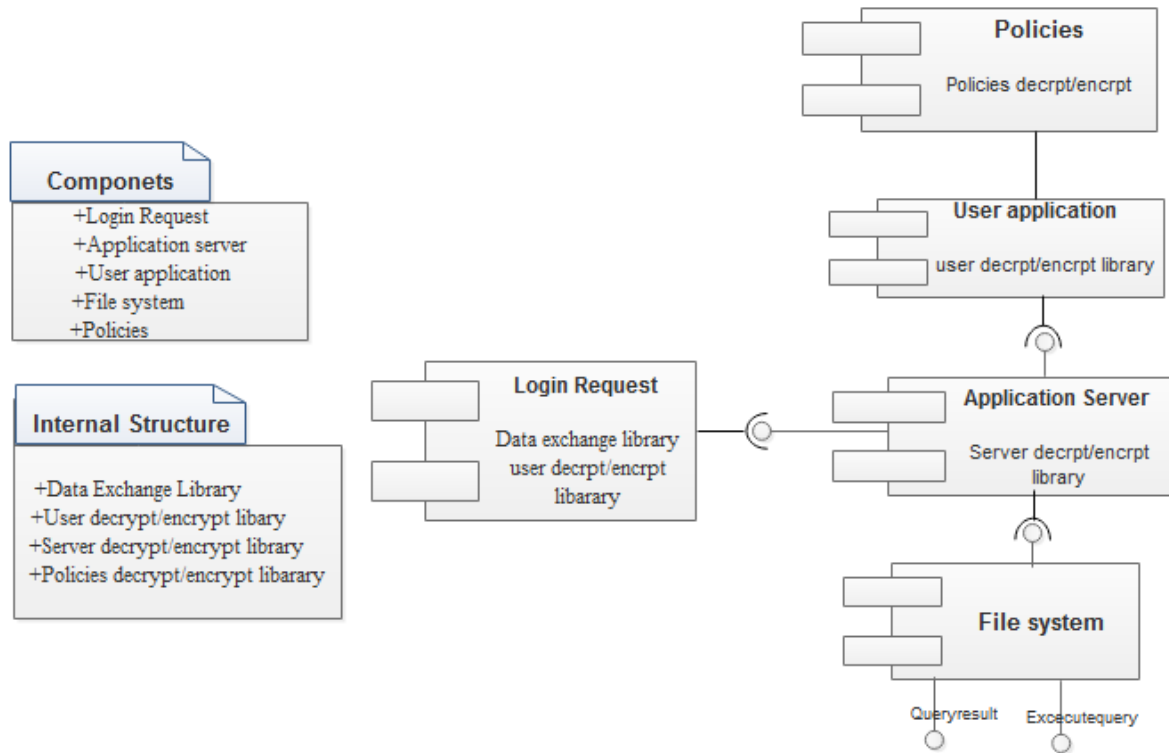
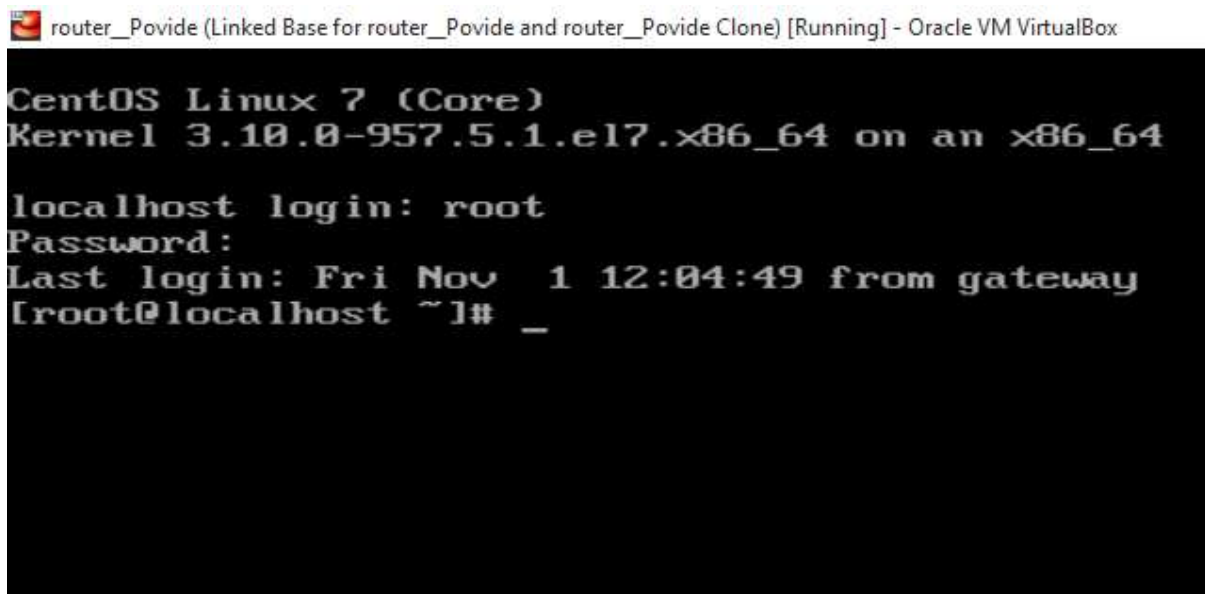


Figure 62: Component Diagram (Author, 2019)

4.8 Model Development

POVIDE Model is hosted in a virtual box which is a free and open source Cloud solution. POVIDE Model was deployed with local storage, called a shared Network File System (NFS) storage. The CLI of the server can be accessed; this is useful as there are more management options via the CLI than on the graphical management interface. POVIDE Model development was typically modeled from different interpretations these are Service providers, policies and security. Model development includes integrated development environments, application lifecycle management components and application security testing components. In the service providers' view, SaaS enterprise Virtual box tool with a built-in visual modeler was used. In the policies view survey of the literature was used and as security is concerned IP addresses were used. Model development followed a group of Agile Software Development Process methodology called Rapid application development and Unified

Modeling Language (UML). RAD emphasizes on working software and user feedback over strict planning and requirement recording and deals with lots of testing. UML is a standardized modeling language enabling developers to specify, visualize, construct and document artifacts of a software system. Thus, UML makes these artifacts scalable, secure and robust in execution. UML is an important aspect involved in object-oriented software development.

The image shows a terminal window titled "router_Povide (Linked Base for router_Povide and router_Povide Clone) [Running] - Oracle VM VirtualBox". The terminal output is as follows:

```
CentOS Linux 7 (Core)
Kernel 3.10.0-957.5.1.el7.x86_64 on an x86_64

localhost login: root
Password:
Last login: Fri Nov  1 12:04:49 from gateway
[root@localhost ~]# _
```

Figure 63: POVIDE Model Interface (Author, 2019)

4.9 POVIDE Model Testing

It entailed the involvement of selected domain experts to test the POVIDE Model in the policy violation issues and comment on the extent to which they thought the model represented real situations while utilizing the cloud. The specific individuals who tested the model were chief network administrators. In order to have uniform feedback from the chief network administrators, they were required to fill the evaluation form while testing. They were guided on how to maneuver in the model. The model also shows the number of individuals who accessed the model as shown in figure 64. To determine the relationship between the perception of the ICT experts, the results were fitted using Poisson regression in

Log Linear Model with the response variable as the count and the distribution being Poisson while the link function was algorithm; $\eta = \log(\mu)$. The outputs are presented in Table 8 and Table 9.

Table 8: Criteria for Assessing Goodness of Fit (Author's, 2019)

Criterion	DF	VALUE	VALUE/DF
Deviance	8	21.8031	2.7254
Scale Deviance	8	21.8031	2.7254
Pearson Chi-square	8	19.994	2.4993
Scaled Pearson X^2	8	19.994	2.4993
Log likelihood		1344.49	

Table 9: Analysis of Parameter Estimates (Author's, 2019)

Parameter	DF	Estimate	Standard Error	Wald 95 confidence		CHISQ	P>CHISQ
				Lower limit	Upper limit		
Intercept	1	3.8067	0.0563	3.9171	3.9171	4564.57	<0.0001
Perception	1	0.1446	0.0979	-0.0473	0.3365	2.18	0.14
Perception	0	0	0	0	0		
Scale	0	1	0	1	1		

The analysis indicates that the perception of the expert does not depend on the ranks the expert gives on the model. This shows that there is an independent association between the perception and the rank of the model by the expert. We, therefore, accept their assessment that where they agreed that the model detects violation of the computer.

[8]	[03703]	[]	[]	[pts/1	[]	[0.0.0.0	[]	[Tue Apr 30 09:37:05 2019 EDT]
[7]	[03850]	[ts/0]	[root	[pts/0	[gateway	[10.0.2.2	[]	[Tue Apr 30 09:38:06 2019 EDT]
[7]	[03871]	[ts/1]	[root	[pts/1	[gateway	[10.0.2.2	[]	[Tue Apr 30 09:38:25 2019 EDT]
[8]	[03867]	[]	[]	[pts/1	[]	[0.0.0.0	[]	[Tue Apr 30 09:47:45 2019 EDT]
[8]	[03846]	[]	[]	[pts/0	[]	[0.0.0.0	[]	[Tue Apr 30 09:47:50 2019 EDT]
[7]	[03954]	[ts/0]	[root	[pts/0	[gateway	[10.0.2.2	[]	[Tue Apr 30 09:48:45 2019 EDT]
[7]	[03975]	[ts/1]	[root	[pts/1	[gateway	[10.0.2.2	[]	[Tue Apr 30 09:49:00 2019 EDT]
[2]	[00000]	[~]	[reboot	[~]	[3.10.0-957.5.1.e17.x86_64	[0.0.0.0	[]	[Wed May 01 05:50:35 2019 EDT]
[5]	[02748]	[tty1]	[]	[tty1	[]	[0.0.0.0	[]	[Wed May 01 05:51:02 2019 EDT]
[6]	[02748]	[tty1]	[LOGIN	[tty1	[]	[0.0.0.0	[]	[Wed May 01 05:51:02 2019 EDT]
[1]	[00051]	[~]	[runlevel]	[~]	[3.10.0-957.5.1.e17.x86_64	[0.0.0.0	[]	[Wed May 01 05:51:30 2019 EDT]
[7]	[02748]	[tty1]	[root	[tty1	[]	[0.0.0.0	[]	[Wed May 01 05:55:57 2019 EDT]
[7]	[03731]	[ts/0]	[root	[pts/0	[gateway	[10.0.2.2	[]	[Wed May 01 06:00:45 2019 EDT]
[7]	[03778]	[ts/1]	[root	[pts/1	[gateway	[10.0.2.2	[]	[Wed May 01 06:02:20 2019 EDT]
[8]	[03699]	[]	[]	[pts/1	[]	[0.0.0.0	[]	[Wed May 01 06:11:10 2019 EDT]
[7]	[03979]	[ts/2]	[root	[pts/2	[gateway	[10.0.2.2	[]	[Wed May 01 06:12:55 2019 EDT]
[2]	[00000]	[~]	[reboot	[~]	[3.10.0-957.5.1.e17.x86_64	[0.0.0.0	[]	[Wed May 29 07:23:37 2019 EDT]
[5]	[02744]	[tty1]	[]	[tty1	[]	[0.0.0.0	[]	[Wed May 29 07:24:03 2019 EDT]
[6]	[02744]	[tty1]	[LOGIN	[tty1	[]	[0.0.0.0	[]	[Wed May 29 07:24:03 2019 EDT]
[1]	[00051]	[~]	[runlevel]	[~]	[3.10.0-957.5.1.e17.x86_64	[0.0.0.0	[]	[Wed May 29 07:24:29 2019 EDT]
[8]	[02744]	[tty1]	[]	[tty1	[]	[0.0.0.0	[]	[Wed May 29 07:26:02 2019 EDT]
[5]	[03663]	[tty1]	[]	[tty1	[]	[0.0.0.0	[]	[Wed May 29 07:26:02 2019 EDT]
[6]	[03663]	[tty1]	[LOGIN	[tty1	[]	[0.0.0.0	[]	[Wed May 29 07:26:02 2019 EDT]
[7]	[03663]	[tty1]	[root	[tty1	[]	[0.0.0.0	[]	[Wed May 29 05:28:32 2019 EDT]
[7]	[03737]	[ts/0]	[root	[pts/0	[gateway	[10.0.2.2	[]	[Wed May 29 05:45:29 2019 EDT]
[7]	[03819]	[ts/1]	[root	[pts/1	[gateway	[10.0.2.2	[]	[Wed May 29 05:57:34 2019 EDT]
[8]	[03815]	[]	[]	[pts/1	[]	[0.0.0.0	[]	[Wed May 29 05:58:40 2019 EDT]
[7]	[03852]	[ts/2]	[root	[pts/2	[gateway	[10.0.2.2	[]	[Wed May 29 05:59:59 2019 EDT]
[8]	[03733]	[]	[]	[pts/0	[]	[0.0.0.0	[]	[Wed May 29 06:03:27 2019 EDT]
[7]	[03913]	[ts/0]	[root	[pts/0	[gateway	[10.0.2.2	[]	[Wed May 29 06:04:01 2019 EDT]
[2]	[00000]	[~]	[reboot	[~]	[3.10.0-957.5.1.e17.x86_64	[0.0.0.0	[]	[Sun Jun 16 03:03:35 2019 EDT]
[5]	[02724]	[tty1]	[]	[tty1	[]	[0.0.0.0	[]	[Sun Jun 16 03:03:47 2019 EDT]
[6]	[02724]	[tty1]	[LOGIN	[tty1	[]	[0.0.0.0	[]	[Sun Jun 16 03:03:48 2019 EDT]
[1]	[00051]	[~]	[runlevel]	[~]	[3.10.0-957.5.1.e17.x86_64	[0.0.0.0	[]	[Sun Jun 16 03:03:53 2019 EDT]
[7]	[03905]	[ts/0]	[root	[pts/0	[gateway	[10.0.2.2	[]	[Sun Jun 16 04:09:05 2019 EDT]
[7]	[03973]	[ts/1]	[root	[pts/1	[gateway	[10.0.2.2	[]	[Sun Jun 16 04:13:58 2019 EDT]
[2]	[00000]	[~]	[reboot	[~]	[3.10.0-957.5.1.e17.x86_64	[0.0.0.0	[]	[Sun Jun 16 07:53:59 2019 EDT]
[5]	[02728]	[tty1]	[]	[tty1	[]	[0.0.0.0	[]	[Sun Jun 16 07:54:12 2019 EDT]
[6]	[02728]	[tty1]	[LOGIN	[tty1	[]	[0.0.0.0	[]	[Sun Jun 16 07:54:12 2019 EDT]
[1]	[00051]	[~]	[runlevel]	[~]	[3.10.0-957.5.1.e17.x86_64	[0.0.0.0	[]	[Sun Jun 16 07:54:15 2019 EDT]
[7]	[02728]	[tty1]	[root	[tty1	[]	[0.0.0.0	[]	[Sun Jun 16 07:56:17 2019 EDT]
[2]	[00000]	[~]	[reboot	[~]	[3.10.0-957.5.1.e17.x86_64	[0.0.0.0	[]	[Mon Jun 24 08:29:42 2019 EDT]
[5]	[02724]	[tty1]	[]	[tty1	[]	[0.0.0.0	[]	[Mon Jun 24 08:29:54 2019 EDT]
[6]	[02724]	[tty1]	[LOGIN	[tty1	[]	[0.0.0.0	[]	[Mon Jun 24 08:29:54 2019 EDT]

Figure 64: List of Logins (Author, 2019)

4. 10 High-Level Overview of the POVIDE Model

The section presents a high-level overview of the POVIDE model. Policy Violation Detection model is a well-defined recurring process model that has been used in a step-by-step approach to scientifically plan and prepare the cloud for policy violation. Moreover, the POVIDE model has been represented as a positive process, which means it deals with real-time detection strategies. The high-level POVIDE model is divided into four distinct tiers as shown in Figure 65, which enables communication between the other processes. The tiers include Virtualization tier, application tier, file system tier, and policy tier.

4.10.1 Virtualization Tier

The virtualization tier uses a hypervisor VMs to provide the user with an interface for connecting with a backend database through a virtual box. The virtual box acts as a thin client which executes only a few of the application logic.

4.10.2 Application Tier

The application tier processes the various inputs and selections received by the virtual machines. It contains a web server that hosts the server application and its supporting components.

4.10.3 File System Tier

The file system tier comprises virtual databases through which applications can be blocked if violations occur. The file system contains user login credentials and log files.

4.10.4 Policy Tier

The policy tier contains the existing detection tools weaknesses used to develop the POVIDE Model. It also verifies if there is a violation when accessing the application in the cloud. In cases where there is a violation, it automatically detects the violation in real time and if there is no violation the connection is established in the cloud.

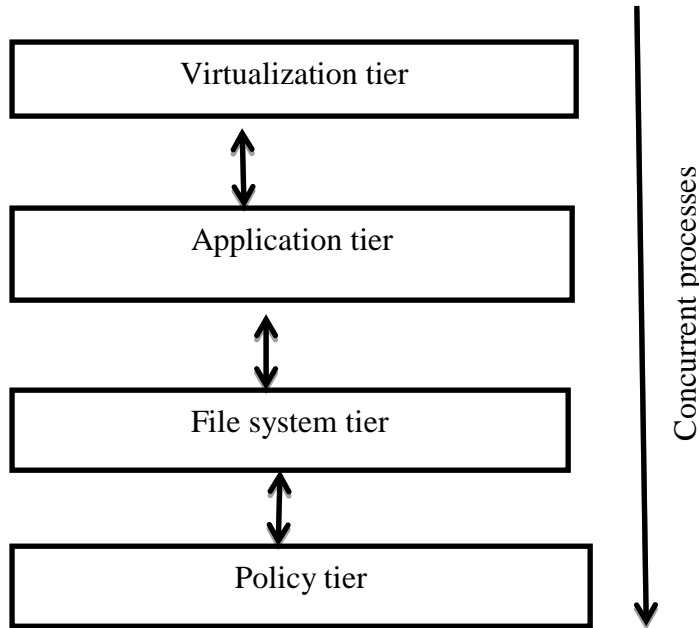


Figure 65: High-level Overview of the POVIDE Model (Author, 2019)

4.11 Experiment Purpose and Scenarios Used

This experiment was meant to precisely prepare a cloud environment for policy violation detection through the collection of common weaknesses with existing detection and prevention tools. This experiment identifies intruders who use vulnerabilities to attack. The experiment focused on demonstrating the weaknesses of firewall and IDS and then developing a more robust tool to curb the weaknesses. The experiment was done as case scenarios that depict attacks in the cloud. This experiment was conducted to test five case scenarios that were elaborated earlier in the chapter.

Scenario one is titled scanning attack, which is when the internal user who is on the network tries to scan the entire network for the opened ports. The virtual machine can suffer from a data breach and theft of personal data. The stolen data can be maliciously used to defraud users in different organizations. This is depicted in a situation where a disgruntled employee exploited the company network server by scanning the entire network setting up a VM and managing to steal confidential information from the cloud environment.

Scenario two is entitled penetration attack, which is when the external attacker who is not on the network manages to attack a virtual machine by collecting all the information and even doing what he/she wants. Here the attacker can take a screenshot of what the user is doing. This happens when a hacker wants to get data from an organization that is confidential or wants to blackmail the user. This is painted when Mark a former network administrator set service in the cloud to retaliate against the earlier actions of eyewitnesses of junior employees V and Y who had affirmed against him in a situation that led to his termination. Mark sends a download to V and he runs the file on his machine not knowing the content and where the source and Mark manage to get the IP address of the machine. This enabled Mark to get hold of their unique router IP address. After this, Mark manages to access his information and blackmailing him.

Scenario three is entitled false positive, is when the intrusion detection system sends the alerts even when the attacks have not happened. It makes the system not to be efficient. This happens when a user is accessing a, for example, a website on the internet or any file but the system is reporting it as an attack. It can even send so many alerts, which others are true attacks but the network administrator would assume. Mary an employee of a certain organization, access a website with this URL <http://www.testmyids.com/> but the server reports the IP address as a live attack.

The fourth scenario is titled false negative, which is when the intrusion detection system cannot detect an attack when it happens. This happens when a machine on the network but the server cannot detect the attacking tool. There is company X and an authorized IP address attacks the machine of the use on the network and downloads files without being detected by the Intrusion detection system.

4.12 Execution of the Model

The experiment was executed by using weaknesses to deploy a modified form of policy violation detection in the cloud environment. The aim was for it to scan the network and block the open ports, which could be vulnerable to the attacker, and to detect and terminate the rogue hosts. It would also test claims that were highlighted in the scenarios presented above. POVIDE Model was deployed from the virtual box server to VMs to gather proofs. The researcher developed a software prototype that was used as intrusion detection and the virtual machine was used. The developed software prototype was able to run an application. This prototype was implemented to test the possibility of detecting policy violations from a simulated cloud environment by modifying the functionality.

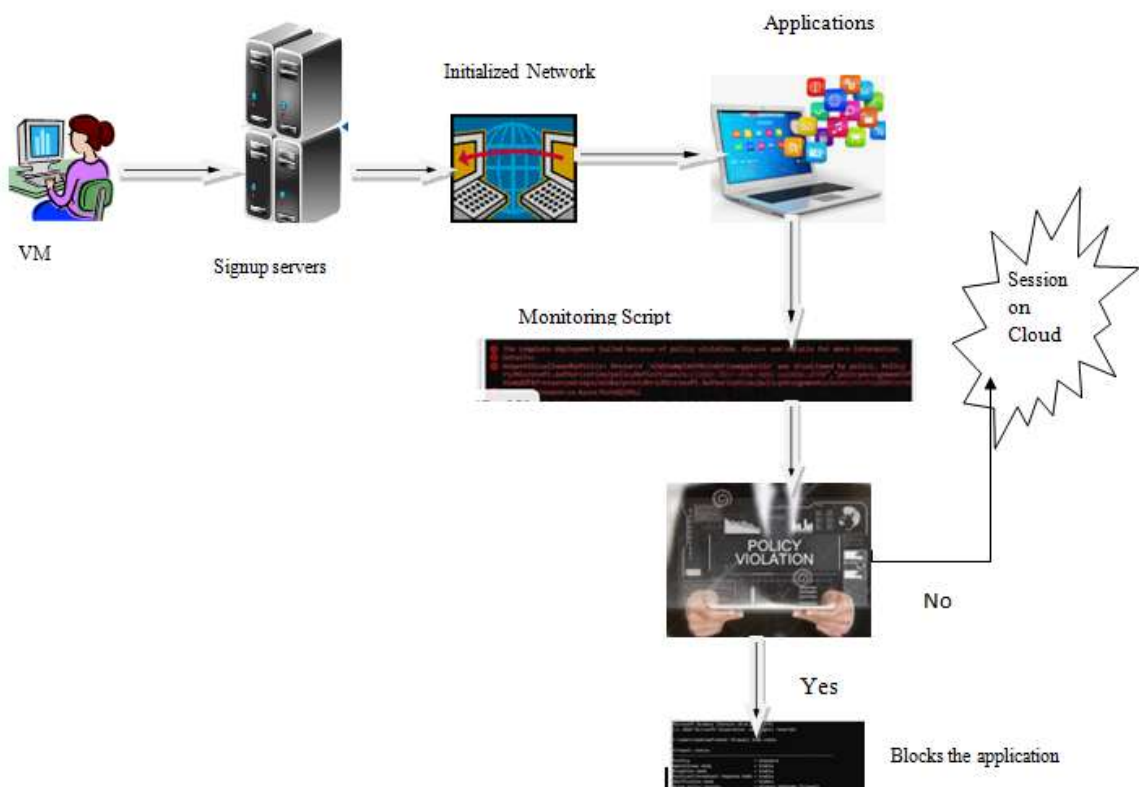


Figure 66: POVIDE Model Execution

4.13 Evaluation

The aim of this research is to evaluate the POVIDE Model recovered from a private Cloud using existing intrusion detection tools. To achieve this aim, structural studies on the Policy Violation detection model was conducted to provide a baseline for evaluating the policy violation value in the cloud. This aspect of the evaluation was purely technical in nature and sought to examine how robust the solution was. Four evaluation measures, model validity, correctness, consistency, and completeness were adopted and used to evaluate the performance of the model.

4.13.1 Criteria

This section critically evaluates the POVIDE Model that was developed by the researcher and used modified functionalities of snort and firewall to develop the model in the cloud environment. The prototype presented in chapter 3 was proof of concept.

To determine this measure, a review was undertaken of the existing weaknesses of intrusion detection and prevention systems. This was discussed in Chapter 3, Section 3.8.1. It was deemed appropriate to use a set of criteria to assess the model. The weaknesses that were proffered by Hock and Kortis (2015); Chowdhary et al., (2014) and the criteria for evaluating POVIDE Model were reviewed in order to identify a set of criteria, which can be used to evaluate weaknesses of existing detection, and prevention tools. Morris (2013) set out the evaluation process. The criteria are as shown in table 10

Table 10: Evaluation Criteria (Author, 2019)

Criteria	Explanation
Validity	It was possible to demonstrate the weaknesses of the existing detection and prevention tools in a way that cannot easily be disputed as demonstrated.
Correctness	It was possible to show that the model that demonstrated the weaknesses is in proper working condition and that the techniques used to develop the model are acceptable within the context of the study as demonstrated.
Consistency	It was possible to demonstrate that the justifiable method was used to test and achieve results as demonstrated
Completeness	It was possible to show that the maximum amount of proof required for the demonstration was done and analyzed as demonstrated

4.13.1.1 Validity

In terms of validity, it should be possible to demonstrate the weaknesses of the firewall and intrusion detection system alongside the Model to curb the weaknesses. It should also demonstrate the procedures and circumstances by which it was tested in a way that cannot easily be disputed. There are three aspects to this: weaknesses; the processes and circumstances that produced the proof; and the fact that it should be indisputable. In the Cloud, existing performances can be used for all three of these aspects. For example, as the model keep a record of network traffic of a user's activities; they can be used to identify the origin of threats and to identify the processes that produced the threats. Given that the POVIDE Model VMs handles the logging, this proof should be indisputable.

In the context of this research, POVIDE Model is able to the identity of the owner of that VM that causes vulnerability to the cloud. The origin of the VM was demonstrated by

viewing the audit logs and IP Address where the actions of each user are recorded, including the virtual machine name and where it's coming from. Therefore, the validity of the model using existing tools' weaknesses can be demonstrated using the audit logs. The origin of weaknesses and the processes that produced the proof can also be demonstrated in this way. Finally, it can also be shown that the audit log cannot easily be changed as only two types of users can access it, therefore the proof is indisputable.

4.13.1.2 Correctness

This criterion stated that it should be possible to show that the machine that created the Policy violation detection Model is in proper working condition and that the procedures used to develop the model is acceptable. Proper working conditions mean that results were generated by the properly typical operation of the Model. Therefore, the techniques should be repeatable and, if repeated, should produce the same results.

It was shown in Chapter 4, how typical operations of the model generated the results. This was demonstrated when the Model could scan and block ports not to allow the attacker through an open port. To verify the correctness of the model, three different methods were used and, for each method, that was tested the results were the same. This showed that when /port-task and at blocked.txt commands were used it would scan and block the open ports. The artifacts from the two scanning methods were analyzed and the results were the same. Therefore, the correctness of the PROVIDE Model was tested from weaknesses of the existing tools such as firewall and intrusion detection tools.

4.13.1.3 Consistency

The criterion of consistency states that it should be possible to demonstrate that the justifiable method was used to obtain the result of the PROVIDE Model. In addition, it states

that existing methods, including existing detection tools, can be shown to be sufficient by acquiring weaknesses needed in a manner that does not compromise either the integrity or the admissibility of the developed model. In order for the results to be considered consistent, it should meet the conditions of this criterion, which effectively means that the results must be reliable. In terms of this research, it was demonstrated in Chapter 4, that existing tools have the capability to detect intrusion in the cloud but they have weaknesses. Two methods were used: one was scanning the network and the other one was penetration to the network. Each method successfully demonstrated the results as required and the analysis of the Model produced predictable results. Therefore, the consistency of the POVIDE Model demonstrated how to curb the weaknesses of the existing detection tools.

4.13.1.4 Completeness

The last criterion states that it should be possible to show that a number of weaknesses required for the demonstration have been collected and analyzed. Another legal requirement for results is that it should be complete (Reilly et al., 2010). Once the results have satisfied this criterion, then the legal requirement has been met.

In this research, it was shown in Chapter 4, which a number of tests were done and results collected. The scanning and penetration stages were analyzed. The audit log contains collaborative results pertaining to user identity and any actions on the penetration stage. Without the audit log, it would have been difficult to establish ownership of the threat. Therefore, the completeness of the results on the POVIDE Model using existing detection tools can be shown by determining the weaknesses of firewalls and IDS.

In terms of the final legal requirement, which stated that results should be believable, there was no single criterion equivalent. However, a combination of the validity, correctness and consistency criteria would provide reassurance of this requirement having been met. This

stated that the collected results represent the facts. Overall, the discussion in this section has demonstrated the results of the POVIDE Model be valid, correct, consistent and complete.

4.14 Comparison of Existing Detection Systems Versus POVIDE Model

Here the researcher is comparing the existing detection systems against the developed model as shown in table 11

Table 11: Comparison of Existing Detection System versus POVIDE Model (Author’s, 2019)

	IDS	Firewall	IPS	POVIDE
General features	Detects attacks but cannot prevent the attacks	Cannot react to a network nor initiate effective countermeasures	Prevent attacks but cannot detect attacks	Detects and prevents attacks in real time
Anomaly Detection	Detects known attacks	Detects known attacks	Cannot detect known attacks but prevent known attacks	Detects both known and unknown attacks
Signature Detection	Cannot detect unknown signatures	Cannot detect unknown signatures	Cannot detect the unknown signature	Detects both known and unknown signature
The network administrator is required to probe the attacks once it is detected	yes	yes	yes	No
Detects Internal attacks	Can Detect	Cannot Detect	Cannot Detect	Can Detect
False Positive	Produces high false positive	Produces high false positive	Produces high false positive	Prevents false positive
Encrypted packets	Encrypted packets are not processed	Encrypted packets are not processed	Encrypted packets are not processed	Detects and blocks both encrypted and decrypted packets

CHAPTER FIVE

SUMMARY, CONCLUSION, AND RECOMMENDATIONS

5.1 Introduction

This chapter presents the summary based on objectives refer to chapter one, literature review in finding the survey of literature as referred to chapter two, it is based on research design methodology and results in chapter three and four respectively. The conclusion also based on objectives findings and recommendations from the study that comprised a survey of literature refer to chapter two, model development, prototype development, and piloting and performance evaluation in chapter three and four. This chapter summarizes this research before drawing up its conclusions, recommendation, and influences to knowledge, and underlining possible future work

5.2 Summary

The summary gives a brief overview of the findings based on each objective. The first objective was to identify weaknesses in the existing intrusion detection and prevention tools mainly IDS and firewalls. The research indicated that the so many weaknesses in regard to firewalls and IDS that other researchers have not tackled that need more research to be done. It also indicated that weak policies contributed to the intrusion detection tools to be weak.

The second objective demonstrated the weaknesses of IDS/ IPS and firewalls on policy violations in the cloud. The objective was achieved and research indicated the weaknesses of IDS/ IPS and firewall using open ports.

The third objective was to develop a model to detect and identify policy violations in real time traffic. The model developed was able to detect the policy violation and block the host trying to violate the policy. POVIDE Model makes a significant impact and creates

healthy competition among Cloud providers to satisfy their Service Level Agreement (SLA) and improve their Quality of Services (QoS). It is important to note that as stated by Becker and Bailey (2014) no one framework or model encompasses all of the possible IT controls, collectively they cover the what, how, and scope of IT Governance. Cloud computing offers many opportunities to organizations, but risks and challenges as well. For an organization to succeed institutions must critically examine available data, create policies especially security policies in the cloud, follow existing standards and develop adequate procedures of ensuring adherence. The objective was achieved by the model's ability to detect policy violations and to implement cloud solutions in a more secure way. Though it is not exhaustive an approach that is oriented on most of the stages that an organization must go through to achieve a relatively secure cloud environment.

The fourth objective evaluated the performance of the model in curbing the weaknesses of IDS and firewalls on policy violations in the cloud. The objective was achieved and research indicated that the model is valid, correct, consistent and complete and with a little improvement would help the cloud in curbing policy violation.

5.3 Conclusions

These days, cloud computing is being distinct and talked about across the ICT industry under special contexts and with different definitions attached to it. Rapid and consistent connectivity is a must for the existence of cloud computing. Cloud computing is without a doubt one of the most enticing technological areas of the current times due, at least in part to its cost-efficiency and flexibility. However, in spite of the flow in activity and interest, there are significant, persistent concerns about cloud computing that are hindering the momentum and will eventually compromise the vision of cloud computing as a new IT procurement model. This concern is regarding privacy and security. Based on the main

objective, four specific Research Objectives were identified. These were to identify weaknesses in the existing intrusion detection and prevention tools mainly IDS and firewalls. To demonstrate the weaknesses of IDS and firewalls on policy violations in the cloud, to develop a model to detect and identify policy violations in real time traffic, To evaluate the performance of the model in curbing the weaknesses of IDS and firewalls on policy violation in the cloud. Based on these objectives, three experiments were tested and carried out in order to generate the data that was needed to test the Model. The model that was selected as the basis for these experiments was PROVIDE Model.

The emphasis of the research was Cloud computing environment, which offers users access to computing resources that can be hosted at the premises of the Cloud Service Provider (CSP) or in remote locations. It offers benefits like cost-saving, convenience, and scalability but it is not without challenges, especially in terms of security, as it can be influenced by criminal activities, as indicated by the Cloud Security Alliance (CSA). Increasing the cost of cybercrime and the growing adoption of Cloud by organizations has demonstrated that there is a need for a policy violation detection model in the cloud environment. However, the architecture of the Cloud, where computing resources are shared together, along with its multi-tenancy and the ease with which resources are released and reallocated, all contribute to making policy violations. This refers to a procedure such as weakness identification, demonstration, experimentation, and evaluation. More positively, the resources offered by the Cloud can be influenced not only for illegal purposes but also for weak policies.

Another way of influencing Cloud resources for the purposes of policy violation is by either adding a detection tool, new or existing, to the Cloud. This is also a step towards achieving a robust model in this cloud environment. To date, research has concentrated on developing tools for specific Cloud technologies with little work on using existing tools. This

research gap formed the basis of this research, which aimed to develop a policy violation detection model in the cloud environment based on the weaknesses of the existing detection tools.

To meet the first objective, a study of a survey of literature concerning the weaknesses of the firewall and intrusion system was undertaken. Various weaknesses on IDS such as false positive and false negative were discussed. Weakness on firewalls such as firewall not detecting internal attacks and unknown attacks were also discussed. A survey of the literature review was conducted to document the weaknesses of the firewall and intrusion detection system. This verified the findings of the literature review in terms of weaknesses. It was found that much has not been done in regard to curbing those weaknesses in the cloud environment.

For the second objective, the focus was on demonstrating the weaknesses of firewalls and IDS on policy violations. Therefore, the first weakness of the firewall was demonstrated through an experiment. The test was done on the firewall to demonstrate if an internal user on the network can scan the network. The weakness was achieved. The second test was on the penetration of external users using a Metasploit exploit tool to gain access to the user's machine by sending a download file. The second weakness of IDS/IPS was demonstrated through the false positive and false negative. False positive was demonstrated by a user's virtual machine accessing a website while the IDS report it as an attack. The other demonstration was on false negative where the external user managed to gain access to the user's machine without being detected by IDs. The experiments demonstrated and verified the weaknesses of firewall and IDS, but went further than this to determine it in relation to the cloud environment. It was found that there are two stages of testing that is penetration and scanning stage. There is a required tool to curb the weaknesses through scanning and detecting true attacks.

In terms of the third objective, experiments were carried out to detect and identify policy violations in real time. The developed model was designed to detect malicious action and terminate the node in real time from a cloud environment as a way of planning and preparing for potential security incidents. If implemented and policies are strict it can work well. It also analyses logs on what is being accessed. The results showed that a virtual machine accessing the network must be scanned for the open ports then in case of any malicious act happens the model detects and terminates the system until verifies by the network administrator.

The emphasis of the fourth objective was acknowledgment. The model was evaluated in terms of validity, correctness, consistent and complete. The model deal with users with Role-Based Access Control (RBAC) and users are authenticated via the Active Directory (AD). PROVIDE Model keeps a log file and monitors the network traffic, which records all user logins. It also scans the network. An experiment was conducted to test if all components work together. The experiment provided the user's activities, which proved that the information in the log can be used to associate data with a particular user.

5.4 Recommendations

- i. Users of the cloud should be trained on the use of cloud and they should be aware of the cloud in their institutions or organizations. Most users used the cloud in their organization but they have limited knowledge on it. The organization staff should be trained so that violation of policies due to the unknowingly accessing malicious applications should not rise.
- ii. Causes of policy violation should be communicated to users and penalties for violation should be put in place to enable self-discipline. Causes of policy violation

should be document and penalties outlined in the staff book so that policy violations should not increase.

- iii. The cloud policies should be outlined properly in each and every organization's policy book. Organizations should use ISO 27001-policy standard as indicated by Council-CSCC, 2012. ISO 27001 provides standards that the organizations can be adapted to allow a smooth use of the cloud.
- iv. Cloud computing technology is changing very fast requiring security measures and policies to be modernized frequently at a speed to match the changing behavior of the cloud policy violations. In addition, licensing is vital to the security of clouds, it would only allow qualified organization rights to host cloud.

5.4.1 Policy recommendations

Policies should be strictly implemented in clouds. Organizational and governing bodies should visit clouds' staff and student's infrastructure on a regular base to evaluate the efficiency of the security precautions adopted by the suppliers. This would minimize the policy violation in the organization because all members of staff would be aware of policies in place.

The government should allow cloud service providers and institutions to incorporate and punish users who violate policies as a breach of contract. Because the government lacks policy standards on the cloud it is very difficult to punish the policy violators leaving different organizations using the cloud at risk.

The government must have national cloud policy, laws and standardized SLA to prevent cloud clients from exploitation since CSP has an upper hand and secretion in implementing the SLA.

Most of the attacks occurring in the cloud must be openly available to determine the reliability of cloud suppliers. This form of sharing helps other network administrators to safeguard against new attacks. It is important to frequently inspect various cloud security-related factors including risks, threats, challenges, vulnerabilities, and attacks. The likelihood of a threat attack can be reduced by extremely understanding the dependencies among other considerations.

Lastly, note that virtualization is the determinant of cloud computing. Nevertheless, the idea of using virtualization in cloud computing is not yet more established as there are several numbers of attacks that target the virtualization environment. Consequently, it is very important to develop reliable schedulers that, by design, contain adequate security mechanisms. The researcher has identified a few areas that are still not covered in the cloud environment security such as appraisal, and migration of data from one cloud to another.

5.4.2 Recommendation for Future work

In the study, objectives that were outlined were achieved based on the suggestions presented in previous chapters. However, the researcher was able to identify a number of issues and challenges hence the researcher suggested the following as topics of research to be conducted in the future. One of the biggest concerns is associated with data; data movement from one cloud to another. Furthermore, clouds fail short of tools that guarantee that the user's data has been deleted from the cloud if the contract is concluded.

In this research, the achievement of the PROVIDE Model is limited to a simulated cloud environment as opposed to a fully-fledged cloud environment. This is because the researcher perceived that the model might be very sensitive to the local laws of a given jurisdiction. Since a variety of standards and policies are not very clear.

The current research focused on scanning and detecting threats that are related to potential crimes; however, due to the increase of technological devices, one may want to know how to deal with the combinatorial explosion of big data. It is suggested that a further extension of the prototype should be developed to locally cache big data at remote and to enable more efficient threat detection.

At the time of this research, there existed no policy standards that had its focus on the cloud environment. It is the researcher's opinion that more research should be conducted so that the PROVIDE Model can be refined further. Additionally, it would be realistic to improve the methodologies and techniques used in developing the prototype with a view to its possible adjustment. In as much as the model could prove the concept the researcher believes that more research, centered in a technical organizational setting, might further highlight the viability of the PROVIDE Model.

Nevertheless, much work remains to be done on the cloud challenges. As the cloud continues to gain power, numerous technical, legal and operational challenges are being experienced. The researcher was able to identify a number of challenges in this research. It is therefore suggested that technical, legal and operational solutions should be pursued that would ensure that the legal aspects of cloud computing keep pace with the advances in the technology.

Lastly, the researcher encouraged a futuristic evaluation of the proposed model, as it would provide an assessment that would help enforce operational tests that may end up providing confidence, based on the performance of the model. Further improvements will allow the model to adapt easily to various scenarios. It is the researcher's opinion that if extensive research on the suggested future work is conducted, then the PROVIDE model adaptation in the cloud environment will be more accepted.

REFERENCE

- Aggarwal, P., & Sharma, S. K. (2015). Analysis of KDD dataset attributes-class wise for intrusion detection. *Procedia Computer Science*, 57, pp. 842-851.
- Agrawal, N., & Tapaswi, S. (2017). The performance analysis of honeypot based intrusion detection system for a wireless network. *International Journal of Wireless Information Networks*, 24(1), pp. 14-26.
- Ahanger, T. A. (2014). Port scan-a security concern. *International Journal of Engineering and Innovative Technology (IJEIT)*, 3(10), pp. 241.
- Ahmed, M., Mahmood, A. N., & Hu, J. (2016). A survey of network anomaly detection techniques. *Journal of Network and Computer Applications*, 60, pp. 19-31.
- AINabulsi, H., Alsmadi, I., & Al-Jarrah, M. (2014). Textual Manipulation for SQL Injection Attacks. *J. of Computer Network and Information Security (IJCNIS)*, 1(1), pp. 26-33.
- Ali, M., Khan, S. U., & Vasilakos, A. V. (2015). Security in cloud computing: Opportunities and challenges. *Information sciences*, 305, pp. 357-383.
- Al-Jarrah, O., & Arafat, A. (2014). "Network Intrusion Detection System using attack behavior classification." In. Information and Communication Systems (ICICS). (2014). 5th International Conference, pp. 1-6.
- Aljawarneh, S., Aldwairi, M., & Yassein, M. B. (2018). Anomaly-based Intrusion Detection System through feature selection analysis and building the hybrid efficient model. *Journal of Computational Science*, 25, pp. 152-160.
- Almorsy, M., Grundy, J., & Müller, I. (2016). An analysis of the cloud computing security problem. *arXiv preprint arXiv:1609.01107*.
- Alnabulsi, H., Islam, M. R., & Mamun, Q. (2014, November). Detecting SQL injection attacks using SNORT IDS. In *Asia-Pacific World Congress on Computer Science and Engineering*, pp. 1-7 IEEE.
- Alsafi, H. M., Abdullaha, W. M., & Pathan, A. S. K. (2012). IDPS: An Integrated Intrusion Handling Model for Cloud. *arXiv preprint arXiv:1203.3323*.
- Alshamrani, A., & Bahattab, A. (2015). A comparison between three SDLC models waterfall model, spiral model, and Incremental/Iterative model. *International Journal of Computer Science Issues (IJCSI)*, 12(1), 106.
- Alshamrani, A., & Bahattab, A. (2015). A comparison between three SDLC models waterfalls model, spiral model, and Incremental/Iterative model. *International Journal of Computer Science Issues (IJCSI)*, 12(1), pp. 106.
- Alsunbul, W. A., Algird, A. R., Sanjer, M. F., & Reddy, K. (2014). Inflatable Device for Intraoperative Control Extension In Cervical Spine Surgery. *Canadian Journal of Neurological Sciences*, 41(2), 293-295.

- Antunes, N., and Vieira, M., (2012).Defending against web application vulnerabilities.ACM, 45, pp. 66–72.
- Ashwini, M. K., Pratiksha, G., Anuja, K., Varsharani, S., & Gayatri, S. (2017). Secure network system using Honeypot. *International Journal of Advanced Research in Computer and Communication Engineering*, 6(2), pp. 230-232.
- Ashwini, M., Gayatri M., & Chawan, P. (2012). Analysis of various software process models. *International Journal of Engineering Research and Applications*, 2(3), pp. 2015-2021.
- Aslahi-Shahri, B. M., Rahmani, R., Chizari, M., Maralani, A., Eslami, M., Golkar, M. J., & Ebrahimi, A. (2016). A hybrid method consisting of GA and SVM for the Intrusion Detection System. *Neural Computing and Applications*, 27(6), pp. 1669-1676.
- Assuncao, M. D., Calheiros, R. N., Bianchi, S., Netto, M. A., & Buyya, R. (2015). Big data computing and clouds: Trends and future directions. *Journal of Parallel and Distributed Computing*, 79, pp. 3-15.
- Avram, M. G. (2014). Advantages and challenges of adopting cloud computing from an enterprise perspective. *Procedia Technology*, 12, pp. 529-534.
- Aziz, T., Razak, A., & Ghani, A. (2017).The performance of different IEEE802: 11 security protocol standards on 2.4 GHz and 5GHz WLAN networks. In: *2017 International Conference on Engineering Technology and Technopreneurship (ICE2T)*, pp. 1-7.
- Bardach, E., & Patashnik, M. (2015). *A practical guide for policy analysis: The eightfold path to more effective problem-solving*. CQ Press.
- Becker, J., & Bailey, E. (2014). A comparison of IT governance & control frameworks in cloud computing.
- Beigi Mohammadi, N., Mišić, J., Mišić, V. B., & Khazaei, H. (2014). A framework for intrusion detection system in advanced metering infrastructure. *Security and Communication Networks*, 7(1), pp. 195-205.
- Bensefia, H., & Ghoualmi, N. (2011). An intelligent system for decision making in firewall forensics. *International Journal of Digital Information and Communication Technology and its Applications*, 166, pp. 470-484.
- Bleikertz, S., Vogel, C., & Groß, T. (2014). Cloud radar: Near real-time detection of security failures in dynamic virtualized infrastructures. In: *Proceedings of the 30th Annual Computer Security Applications Conference* pp. 26-35.
- Buck, K., & Hanf, D. (2010). Cloud SLA considerations for the Government Consumer: Cloudcomputing. *International Conference on Signal Processing, Communication, Power and Embedded System (SCOPE5)*, pp. 527-531.
- Bulajoul, W., James, A., & Pannu, M. (2015). Improving network Intrusion Detection System performance through quality of service configuration and parallel technology. *Journal of Computer and System Sciences*, 81(6), pp. 981-999.

- Bursztein, E., Martin, M., & Mitchell, J.C. (2011). Text-based CAPTCHA strengths and weaknesses. *ACM Conference on Computer and Communications Security*.
- Butun, I., Morgera, S. D., & Sankar, R. (2014). A survey of intrusion detection systems in wireless sensor networks. *IEEE Communications Surveys & Tutorials*, 16(1), pp. 266-282.
- Buyya, R., Yeo, S., Venugopal, S., Broberg, J., & Brandic, I., (2009). Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation computer systems*, 25(6), pp 599-616.
- Carlin, A., Hammoudeh, M., & Aldabbas, O. (2015). Intrusion detection and countermeasure of virtual cloud systems-state of the art and current challenges. *International Journal of Advanced Computer Science and Applications*, 6(6).
- Chang, V. (2015). *A proposed cloud computing business framework*. Nova Science Publisher.
- Chang, V., Walters, R. J., & Wills, G. (2013a). The development that leads to the cloud computing business framework. *International Journal of Information Management*, 33(3), pp. 524-538.
- Chang, V., Walters, R. J., & Wills, G. (2013b). Cloud Storage and Bioinformatics in a private cloud deployment: Lessons for data-intensive research. In. *Cloud computing and service Science*, Springer Lecture Notes Series, Springer Book.
- Chaware, S. (2011). Banking Security using Honeypot. *International Journal of Security and its Applications*, 5(1), pp. 31-38.
- Chen, T., Zhang, X., Jin, S., & Kim, O. (2014). Efficient classification using parallel and scalable compressed model and its application on intrusion detection. *Expert Syst. Appl.* 41(13), pp. 5972–5983
- Chiu, D., Weng, S. H., & Chiu, J. (2017). Backdoor use in targeted attacks. *A Trend Micro Research Paper*.
- Chomsiri, T., He, X., Nanda, P., & Tan, Z. (2014, September). A stateful mechanism for the tree-rule firewall. In *2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications* (pp. 122-129). IEEE.
- Chowdhary, M., Suri, S., & Bhutani, M. (2014). Comparative study of the Intrusion Detection System. *International Journal of Computer Sciences and Engineering*, 2(4), pp. 197-200.
- Council-CSCC, C. S. C. (2012). The CSCC practical guide to cloud service level agreements.
- Dabbour, M., Alsmadi, I., & Alsukhni, E. (2013). Efficient assessment and evaluation for websites vulnerabilities using SNORT. *International Journal of Security and Its Applications*, 7(1), 7-16.
- Dagada, R. (2014). *Legal and policy aspects to consider when providing information security in the corporate environment* (Doctoral dissertation, University of South Africa).

- Dai, J., Gan, Z., Han, B., & Liu, X., (2013). *U.S. Patent No. 8,578,370*. Washington, DC: U.S. Patent and Trademark Office.
- Dall, C., & Nieh, J. (2014). KVM/ARM: The design and implementation of the Linux ARM hypervisor. *ACM SIGARCH Computer Architecture News*, 42(1), pp. 333-348.
- Dash, S. B., Saini, H., Panda, T. C., & Mishra, A. (2014). Service level agreement assurance in cloud computing: A trust issue. *International Journal of Computer Science and Information Technologies*, 5(3), pp. 2899-2906.
- Day, D., & Burns, B. (2011, February). A performance analysis of snort and suricata network intrusion detection and prevention engines. In *Fifth International Conference on Digital Society, Gosier, Guadeloupe* (pp. 187-192).
- Devikrishna, K. S., & Ramakrishna, B. B. (2013). An artificial neural network-based Intrusion Detection System and classification of attacks. *International Journal of Engineering Research and Applications (IJERA)*, 3(4), pp. 1959-1964.
- Dewanjee, R. (2016, October). Intrusion Filtration System (IFS)-mapping network security in new way. In *2016 International Conference on Signal Processing, Communication, Power and Embedded System (SCOPE5)* (pp. 527-531). IEEE.
- Dhopte, S., & Chaudhari, P. M. (2014). Genetic algorithm for Intrusion Detection System. *International Journal of Research in Information Technology (IJRIT)*, 2(3), pp. 503-509.
- Dietrich, N. (2017). Snort 2.9. 9. x on Ubuntu 14 and 16. *línea*]. Available: <https://www.snort.org/documents/snort-2-9-9-x-on-ubuntu-14-16> Date accessed: 13 June 2018.
- Dighe, P., Agrawal, P., Karnick H., Thota S., & Raj B. (2013). Scale-independent raga identification using chromagram patterns and swara-based features. In. *2013 IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*, pp. 1–4.
- Du, P., & Li, X. (2016, October). T-Netm: Transparent network monitoring on virtual machine. In *2016 2nd International Conference on Cloud Computing and Internet of Things (CCIOT)*, pp. 64-67. IEEE.
- Edwards, N., & Dalton, C. I. (2014). *U.S. Patent No. 8,719,914*. Washington, DC: U.S. Patent and Trademark Office.
- Emeakaroha V., Netto M., Calheiros R., Brandic I., Buyya R., and deRose, C., (2012) “Towards autonomic detection of SLA violations in cloud infrastructures,” *Future Generation Computer Systems*, vol. 28, no. 7, pp. 1017–1029.
- Erl, T., Puttini R., & Mahmood Z. (2013). *Cloud computing: Concepts, technology & architecture* (1st Ed.). Massachusetts: Prentice Hall PTG.
- Eronen, P., & Zitting, J. (2001). An expert system for analyzing firewall rules. In: *Proceedings of the 6th Nordic Workshop on Secure IT Systems*, pp. 100–107.

- Fatema K., Vincent C., Emeakaroha P., Healy D., John P., Morrison P., and Lynn T.,(2014)."A survey of Cloud monitoring tools: Taxonomy, capabilities and objectives," 1. *Parallel Distrib Comput.vol.74*, pp. 2918-2933.
- Felici, M., & Pearson, S. (2014). Accountability for data governance in the cloud. In *Summer School on Accountability and Security in the Cloud*, pp. 3-42 Springer, Cham.
- Feng, W., Zhang, Q., Hu, G., & Huang, X. (2014). Mining network data for intrusion detection by combining SVMs with ant colony networks. *Future Generation Computer Systems*, 37, pp. 127-140.
- Fernandez, E. B., Yoshioka, N., & Washizaki, H. (2014). Patterns for cloud firewalls. *Asian PLoP (Pattern Languages of Programs)*, Tokyo.
- Ferreira, J., Soares, J. N., Jardim-Goncalves, R., & Agostinho, C. (2017, May). Management of IoT devices in a physical network. In *2017 21st International Conference on Control Systems and Computer Science (CSCS)*, pp. 485-492 IEEE.
- Fevre, R., Lewis, D., Robinson, A., & Jones, T. (2012). Insight into ill-treatment in the workplace: Patterns, causes, and solutions. *Contemporary Readings in Law & Social Justice*, 4(2).
- Fontugne, R., Mazel, J., & Fukuda, K. (2014). Hadoop: a MapReduce framework for network anomaly detection. In: *2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp 494–499.
- Ford, M., Mallery, C., Palmasani, F., Rabb, M., Turner, R., Soles, L., & Snider, D. (2016). A Process to Transfer Fail2ban Data to an Adaptive Enterprise Intrusion Detection and Prevention System. *Proceedings of the 2016 IEEE Southeast Con*, March 31-April 3, 2016, Norfolk, VA.
- Fujinoki, H. (2013). Dynamic Binary User-Splits to Protect Cloud Servers from DDoS Attacks. Paper presented at the Proceedings of the Second International Conference on Innovative Computing and Cloud Computing, Wuhan, China.
- Galante, J., Kharif, O., & Alpeyev, P. (2011, May 16). Sony Network Breach Shows Amazon Cloud's Appeal for Hackers. Retrieved 2019, from
- Gamage, N. K., Whitner, R. B., & Bartz, T. G. (2016). *U.S. Patent No. 9,270,542*. Washington, DC: U.S. Patent and Trademark Office.
- Ganga, I. S. (2014). *U.S. Patent No. 8,798,056*. Washington, DC: U.S. Patent and Trademark Office.
- Gangwar, H., Date, H., & Ramaswamy, R. (2015). Understanding the determinants of cloud computing adoption using an integrated TAM-TOE model. *Journal of Enterprise Information Management*, 28(1), pp. 107-130.
- Ghorbel, A., Ghorbel, M., & Jmaiel, M. (2017). Privacy in cloud computing environments: A survey and research challenges. *The Journal of Supercomputing*, 73(6), pp. 2763-2800.

- Ghosh, A., Cosby, S., Keister, A., Bryant, B., & Taylor, S. (2018). *U.S. Patent Application No. 10/043,001*.
- Gilchrist, A. (2016). Designing industrial internet systems. In *Industry 4.0* (pp. 87-118). Apress, Berkeley, CA.
- Golshan, A., & Binder, S., (2016). *U.S. Patent No. 9,519,781*. Washington, DC: U.S. Patent and Trademark Office.
- Gong, C., Liu, J., Zhang, Q., Chen, H., & Gong, Z. (2010, September). The characteristics of cloud computing. In: *Parallel Processing Workshops (ICPPW), 2010 39th International Conference on* pp. 275-279 IEEE.
- Gonzales, D., Kaplan, J. M., Saltzman, E., Winkelman, Z., & Woods, D. (2017). Cloud-trust—A security assessment model for infrastructure as a service (IaaS) clouds. *IEEE Transactions on Cloud Computing*, 5(3), pp. 523-536.
- Goodin, D. (2009, December 9). 'Zeus bot found using Amazon's EC2 as C&C server.' The Register.
- Gould, K., & Danforth, A. (2016). *U.S. Patent No. 9,497,503*. Washington, DC: U.S. Patent and Trademark Office.
- Goyal, S. (2014). Public vs private vs hybrid vs community-cloud computing: A critical review. *International Journal of Computer Network and Information Security*, 6(3), 20.
- Gozman, D., & Willcocks, L.P. (2015). Crocodiles in the regulatory swamp: Navigating the dangers of outsourcing, SaaS and shadow IT. *ICIS*.
- Gul I., & Hussain M. (2011). Distributed cloud intrusion detection model. *Int. J. adv. sci. technology* 34 (38) pp. 135.
- Gul, I., & Hussain, M. (2011). Distributed cloud intrusion detection model. *International Journal of Advanced Science and Technology*, 34(38), 135.
- Hameed, A., Khoshkbarforousha, A., Ranjan, R., Jayaraman, P. P., Kolodziej, J., Balaji, P., & Khan, S. U. (2016). A survey and taxonomy on energy-efficient resource allocation techniques for cloud computing systems. *Computing*, 98(7), pp. 751-774.
- Hamidi A., Hadi S., and Sharif M., (2011)."A transparent virtual machine monitor level package compression network service," *Procedia Computer Science*. Vo.3, pp. 401-407.
- Hashem, I. A. T., Yaqoob, I., Anuar, N. B., Mokhtar, S., Gani, A., & Khan, S. U. (2015). The rise of "big data" on cloud computing: Review and open research issues. *Information systems*, 47, pp. 98-115.
- Hatem, S. S., & El-Khouly, M. M. (2014). Malware detection in cloud computing. *Int J Adv Comput Sci Appl*, 5(4), 187-192.

- He, X., Chomsiri, T., Nanda, P., & Tan, Z. (2014). Improving cloud network security using the Tree-Rule firewall. *Future Generation Computer Systems*, 30, pp. 116-126.
- Hock, F., & Kortis, P. (2015). Commercial and open-source based Intrusion Detection System and Intrusion Prevention System (IDS/IPS) design for IP networks. In: *Emerging eLearning Technologies and Applications (ICETA), 2015 13th International Conference on*, pp. 1-4.
- Hu, H., Han, W., Ahn, G. J., & Zhao, Z. (2014). FLOWGUARD: Building robust firewalls for software-defined networks. In *Proceedings of the Third Workshop on Hot Topics in Software-defined Networking*, pp. 97-102 ACM.
- Huang, D., & Wu, H. (2017). *Mobile Cloud Computing: Foundations and Service Models*. (1st Ed.). Morgan Kaufmann.
- Huang, V. S. M., Huang, R., & Chiang, M. (2013, March). A DDoS mitigation system with multi-stage detection and text-based during testing in cloud computing. In *2013 27th International Conference on Advanced Information Networking and Applications Workshops*, pp. 655-662 IEEE.
- Ibrahim, A. S., Hamlyn-Harris, J., & Grundy, J. (2016). Emerging security challenges of cloud virtual infrastructure. In *Proceedings of APSEC 2010 Cloud Workshop*, Sydney, Australia, 30th Nov 2010.
- Indre, I., & Lemnaru, C. (2016). Detection and prevention system against cyber-attacks and botnet malware for information systems and Internet of Things. *IEEE 12th International Conference on Intelligent Computer Communication and Processing (ICCP) (2016)*, pp. 175-182, Cluj-Napoca.
- Jabez, J., & Muthukumar, B. (2015). An Intrusion Detection System (IDS): Anomaly detection using outlier detection approach. *Procedia Computer Science*, 48, pp. 338-346.
- Jaiganesh, V., Sumathi, P., & Mangayarkarasi, S. (2013). An analysis of Intrusion Detection System using backpropagation neural network. *IEEE Computer Society Publication*.
- Jansen, B., & Tanner, A. (2014). *U.S. Patent No. 8,875,272*. Washington, DC: U.S. Patent and Trademark Office.
- Jansen, B., & Tanner, A. (2014). *U.S. Patent No. 8,875,272*. Washington, DC: U.S. Patent and Trademark Office.
- Jing, Q., Vasilakos, A. V., Wan, J., Lu, J., & Qiu, D. (2014). Security of the Internet of Things: perspectives and challenges. *Wireless Networks*, 20(8), pp. 2481-2501.
- Jones, N., George, E., Merida, F.I., Ramussen, U., & Volzow, V. (2014). *Electronic Evidence Guide - A Basic Guide for Police Officers, Prosecutors, and Judges*. Retrieved from https://au.int/sites/default/files/newsevents/workingdocuments/34122-wd-annex_4_-_electronic_evidence_guide_2.0_final-complete.pdf

- Joy, A. M. (2015, March). Performance comparison between linux containers and virtual machines. In *2015 International Conference on Advances in Computer Engineering and Applications* pp. 342-346. IEEE.
- Jula, A., Sundararajan, E., & Othman, Z. (2014). Cloud computing service composition: A systematic literature review. *Expert systems with applications*, 41(8), 3809-3824.
- Kadayiruppu, E. (2014). Investigative analysis of security issues and challenges. In Cloud computing and their counter measures. *Journal Impact Factor*, 5(12), pp. 57-63.
- Kalaiprasath, R., Elankavi, R., & Udayakumar, D. R. (2017). Cloud. Security and Compliance-A Semantic Approach in End to End Security. *International Journal Of Mechanical Engineering And Technology (Ijmet)*, 8(5), pp. 987-994.
- Kambow, N., & Passi, L. K. (2014). Honeypots: The need for network security. *International Journal of Computer Science and Information Technologies*, 5(5), pp. 6098-6101.
- Karnwal T., Sivakumar T., & Aghila G. (2012). A comber approach to protect cloud computing against XML DDoS and HTTP DDoS attack. In Proceedings of the 2012 IEEE Students' Conference pp.2.
- Karthikeyan, R., Geetha, D. T., Shyamamol, K. S., & Sivagami, G. (2017). Advanced Honeypot architecture for network threats quantification. *The International Journal of Engineering and Techniques*, 3(2), pp. 2395-1303.
- Kaur, T., Malhotra, V., & Singh, D. (2014). Comparison of network security tools-firewall, Intrusion Detection System, and Honeypot. *Int. J. Enhanced Res. Sci. Technol. Eng*, pp. 200-204.
- Kavis, M. J. (2014). *Architecting the cloud: Design decisions for cloud computing service models (SaaS, PaaS, and IaaS)*. John Wiley & Sons.
- Kene, S. G., & Theng, D. P. (2015). A review of intrusion detection techniques for cloud computing and security challenges. In *Electronics and Communication Systems (ICECS), 2015 2nd International Conference on* pp. 227-232. IEEE.
- Keshri, A., Singh, S., Agarwal, M., & Nandiy S. (2016). DoS attack prevention using IDS and data mining. *International Conference on Accessibility to Digital World (ICADW)* pp. 87-92 Guwahat.
- Khalil, I. M., Khreishah, A., & Azeem, M. (2014). Cloud computing security: A survey. *Computers*, 3(1), pp. 1-35.
- Khamphakdee, N., Benjamas, N., & Saiyod, S. (2014, May). Improving intrusion detection system based on snort rules for network probe attack detection. In *2014 2nd International Conference on Information and Communication Technology (ICoICT)* pp. 69-74. IEEE.
- Khurana, H., Guralnik, V., & Shanley, R. (2014). *U.S. Patent No. 8,793,790*. Washington, DC: U.S. Patent and Trademark Office.

- Kim, D., & Solomon, M. G. (2016). *Fundamentals of information systems security*. Jones & Bartlett Publishers.
- Kirat, D., Vigna, G., & Kruegel, C. (2014). BareCloud: Bare-metal analysis-based evasive malware detection. In *USENIX Security Symposium*, pp. 287-301.
- Kondra, J. R., Mishra, S. K., Bharti, S. K., & Babu, K. S. (2016). Honeypot-Based Intrusion Detection System: A performance analysis. *International Conference on "Computing for Sustainable Global Development"*, INDIA Com (pp. 3947-3951). New Delhi: IEEE.
- Krombholz, K., Hobel, H., Huber, M., & Weippl, E. (2015). Advanced social engineering attacks. *Journal of Information Security and Applications*, 22, pp. 113-122.
- Kumar, M., & Hanumanthappa, M. (2013). Scalable intrusion detection systems log analysis using cloud computing infrastructure. In: *2013 IEEE International Conference on Computational Intelligence And Computing Research (ICCIC)*.pp 1-4.
- Kumar, V., & Sharma, G. D. (2016). Towards configured Intrusion Detection Systems. *Global Journal of Computer Science and Technology*, 16(4), pp. 1-10.
- Larman, C., & Basili, V. R. (2003). Iterative and incremental developments. a brief history. *Computer*, 36(6), pp. 47-56.
- Latha, K. M. (2016). Learn about intrusion detection and prevention. United States: Juniper Networks.
- Lee, K., Kim, D., Ha, D., Rajput, U., & Oh, H. (2015, September). On security and privacy issues of fog computing supported the Internet of Things environment. In: *Network of the Future (NOF), 2015 6th International Conference on the*, pp. 1-3 IEEE.
- Li, C., & Gaudiot, J. L. (2019, July). Detecting Malicious Attacks Exploiting Hardware Vulnerabilities Using Performance Counters. In *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)* Vol. 1, pp. 588-597. IEEE.
- Liao, H., Lin, Y., & Tung, K. (2013). Intrusion Detection System: A comprehensive review, *Journal of New York and Computer Applications*, 36, pp. 16-24.
- Lin W., & Lee D. (2012). Trace back attacks in cloud Pebbletrace botnet. In Proceedings of the 2012 32nd International Conference on Distributed Computing Systems Workshops (ICDCSW), Macau, China, pp. 417-426.
- Linora, J. A., & Barathy, M. N. (2014). Intrusion detection and prevention by using light weight virtualization in web applications. *International Journal of Computer Science and Mobile Computing (IJCSMC)*3(3), pp. 392-396. IEEE
- Liu S., & Chen Y. (2010) Retrospective detection of malware attacks by cloud computing. In Proceedings of the 2010 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), Huangshan, China pp. 510-517.
- Liu, F., Tong, J., Mao, J., Bohn, R., Messina, J., Badger, L., & Leaf, D. (2011). NIST cloud computing reference architecture. *NIST Special Publication, 500*, pp. 1-28.

- Lopez, M. E. A. (2018). *A monitoring and threat detection system using stream processing as a virtual function for big data* (Doctoral dissertation) Universidade Federal do Rio de Janeiro.
- Lo, C. C., Huang, C. C., & Ku, J. (2010, September). A cooperative intrusion detection system framework for cloud computing networks. In *2010 39th International Conference on Parallel Processing Workshops* pp. 280-284. IEEE.
- Lu, X., Yin, J., Xiong, N. N., Deng, S., He, G., & Yu, H. (2016). JTangCMS: An efficient monitoring system for cloud platforms. *Information Sciences*, 370, pp. 402-423.
- Lynn, R. (2011). Investigation of Efficient Unified Threat Management in Enterprise Security.
- Malav, S., Avinash, M. S., Satish, N. S., & Sandeep, S. C. (2016). Network security using IDS, IPS & Honeypot. *International Journal of Recent Research in Mathematics computer Science and Information Technology* 2(2), pp. 27-30
- Marsico, P. J. (2015). *U.S. Patent No. 9,083,680*. Washington, DC: U.S. Patent and Trademark Office.
- Martin, A. (2014, December 24). 'Rackspace knocked offline by huge DDoS attack knocked-offline-huge-DDoS-attack'. *We Live Security*. Retrieved from
- Martinez, F. R., & Pulier, E. (2015). *U.S. Patent No. 9,069,599*. Washington, DC: U.S. Patent and Trademark Office.
- Maynard, P., McLaughlin, K., & Haberler, B. (2014). Towards Understanding Man-In-The-Middle Attacks on IEC 60870-5-104 SCADA Networks. Paper presented at International Symposium for ICS & SCADA Cyber Security Research (ICS-CSR), St Polten, Austria.
- Mazzariello, C., Bifulco, R., & Canonico, R. (2010). Integrating a Network IDS into an Open Source Cloud Computing Environment. *Sixth International Conference on Information Assurance and Security* pp. 265-270.
- McDougal, M. D., Lee, J. J., & Gilmore, W. L. (2014). *U.S. Patent No. 8,914,882*. Washington, DC: U.S. Patent and Trademark Office.
- Mell, P., & Grance, T. (2011). The NIST definition of cloud computing.
- Merlo, A., Migliardi, M., & Spadaccini, E. (2016). Balancing delays and energy consumption in IPS-enabled networks. *2016 30th International Conference on Advanced Information Networking and Applications Workshops (WAINA)* pp. 267-272.
- Milenkoski, A., Vieira, M., Kounev, S., Avritzer, A., & Payne, B., (2015). Evaluating computer intrusion detection systems: A survey of common practices. *ACM Computing Surveys (CSUR)*, 48(1), pp. 12.
- Modi, C.N., Patel, D.R., Patel, A., & Muttukrishnan, R. (2012). Bayesian Classifier and Snort based network intrusion detection system in cloud computing. *2012 Third*

International Conference on Computing, Communication and Networking Technologies (ICCCNT'12), pp. 1-7.

- Modi, C., Patel, D., Borisaniya, B., Patel, H., Patel, A., & Rajarajan, M. (2013). A survey of intrusion detection techniques in Cloud, *Journal of Network and Computer Applications*, 36, pp. 42-57.
- Montes, J., Sánchez, A., Memishi, B., Pérez, M. S., & Antoniu, G. (2013). GMonE: A complete approach to cloud monitoring. *Future Generation Computer Systems*, 29(8), pp. 2026-2040.
- Morris, S.L.A. (2013) *An Investigation into the identification, reconstruction, and evidential value of thumbnail cache file fragments in unallocated space* PhD Thesis. Cranfield University, Shrivenham.
- Mosbah M., Sol, H., Alnashar., & El-Nasr M., (2014). Cloud computing framework for solving Egyptian Higher Education. In: *2014 Fourth International Conference on Advances in Computing and Communications* pp. 208–213, IEEE.
- Mugenda, O. M., & Mugenda, A. G.(2003). *Research methods*.
- Munassar, N. M. A., & Govardhan, A. (2010). A comparison between five models of software engineering. *International Journal of Computer Science Issues (IJCSI)*, 7(5), pp. 94.
- Muthurajkumar, S., Kulothungan, K., Vijayalakshmi, M., Jaisankar, N., & Kannan, A. (2013). A rough set based feature selection algorithm for effective intrusion detection in a cloud model. *Proceedings of the international conference on advances in communication, network, and computing* pp. 8–13.
- Naik, N. (2015, October). Fuzzy inference based intrusion detection system: FI-Snort. In *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*pp. 2062-2067. IEEE.
- Naik, N., & Jenkins, P. (2016, August). Enhancing windows firewall security using fuzzy reasoning. In: *2016 IEEE 14th Intl Conf on Dependable, Autonomic and Secure Computing, 14th Intl Conf on Pervasive Intelligence and Computing, 2nd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech)* pp. 263-269 IEEE.
- Newman, R. C. (2009). *Computer security: Protecting digital resources*. Jones & Bartlett Publishers.
- Noehr, J. (2011) ‘Denial of service attack.’ [Blog] *Bit bucket*. Retrieved from: <https://blog.bitbucket.org/2011/06/06/denial-of-service-attack>.
- Nurika, O., Aminz, A., Rahman, A., & Zakaria, M. (2012). Review of various firewall deployment models. In: *Proceedings of the International Conference on Computer and Information Science*pp. 825–829.

- Ogweno, K. L., Oteyo, O. E., & Henry, D. O. (2014). Honeypot Intrusion Detection System. *International Journal of Engineering Inventions*4(5), pp. 28-41.
- Oliveira, T., Thomas, M., & Espadanal, M. (2014). Assessing the determinants of cloud computing adoption: An analysis of the manufacturing and services sectors. *Information & Management*, 51(5), pp. 497-510.
- Opara-Martins, J., Sahandi, R., & Tian, F. (2014, November). Critical review of vendor lock-in and its impact on adoption of cloud computing. In *International Conference on Information Society (i-Society 2014)* pp. 92-97. IEEE
- Paganini, P. (2014, December 26). 'Lizard Squad took down again Sony PSN and Xbox Live networks. *Security Affairs*. Retrieved from
- Pandeeswari, N., & Kumar, G. (2016). Anomaly detection system in cloud environment using fuzzy clustering based ANN. *Mobile Networks and Applications*, 21(3), pp. 494-505.
- Papp, D., Ma, Z., & Buttyan, L. (2015). Embedded systems security: Threats, vulnerabilities, and attack taxonomy. In *Privacy, Security and Trust (PST), 2015 13th Annual Conference* pp. 145-152 IEEE.
- Patel, A., Taghavi, M., Bakhtiyari, K., & Ju'nior, C. (2013). An intrusion detection and prevention system in Cloud Computing: A systematic overview. *Journal of Network and Computer Applications*, 36, pp. 25-41.
- Patel, K. (2013). Security survey for cloud computing: Threats & existing IDS/IPS techniques, *International Conference on control, Communication and Computer Technology* pp. 5.
- Patel, S. K., & Sonker, A. (2016). Rule-based network intrusion detection system for port scanning with efficient port scan detection rules using snort. *International Journal of Future Generation Communication and Networking*, 9(6), pp. 339-350.
- Pearson, S. (2013). Privacy, security, and trust in cloud computing. In: *Privacy and Security for Cloud Computing* pp. 3-42 Springer, London.
- Pino, R. E., & Kott, A. (2014). Neuromorphic computing for cognitive augmentation in cyber defense. In: *Cybersecurity Systems for Human Cognition Augmentation* pp. 19-45 Springer, Cham.
- Puthal, D., Sahoo, B. P. S., Mishra, S., & Swain, S. (2015, January). Cloud computing features, issues, and challenges: a big picture. In: *Computational Intelligence and Networks (CINE), 2015 International Conference on* pp. 116-123 IEEE.
- Quah, M., Y., & Rohm, U. (2013). User awareness and policy compliance of data privacy in cloud computing. In: *Proceedings of the First Australasian Web Conference-Volume 144*, pp. 3-12. Australian Computer Society, Inc.
- Ramachandran, M., & Chang, V. (2014). Financial software as a service: A paradigm for risk modelling and analytics. *International Journal of Organizational and Collective Intelligence*, 4(3).

- Ramachandran, M., Chang, V., & Li, C. S. (2015). The improved cloud computing adoption framework to deliver secure services. In: *Proceedings of ESaaS 2015-2nd International Workshop on Emerging Software as a Service and Analytics, In conjunction with the 5th International Conference on Cloud Computing and Services Science-CLOSER 2015* pp. 73-79.
- Rashid, F. Y. (2014). 'How hackers target cloud services for Bitcoin profit'. *Security Week*. Retrieved from: <http://www.securityweek.com>.
- Rebahi, Y., Hohberg, S., Shi, L., Parreira, B. M., Kourtis, A., Comi, P., & Ramos, A. (2015, December). Virtual security appliances: the next generation security. In *2015 International Conference on Communications, Management and Telecommunications (ComManTel)* pp. 103-110. IEEE.
- Reilly, D., Wren, C., & Berry, T. (2010) Cloud computing: Forensic challenges for law enforcement. *International Conference for Internet Technology and Secured Transactions (ICITST)*, pp. 1-7.
- Riaz, A., Ahmad, H. F., Kiani, A. K., Qadir, J., Ur Rasool, R. A. I. H. A. N., & Younis, U. (2017). Intrusion Detection Systems in Cloud Computing: A contemporary review of techniques and solutions. *Journal of Information Science & Engineering*, 33(3).
- Ridho, M. F. (2014). *Analysis and evaluation Snort, Bro, and Suricata as an Intrusion Detection System based on Linux server* (Doctoral dissertation: Universitas Muhammadiyah Surakarta).
- Rittinghouse, J. W., & Ransome, J. F. (2016). *Cloud Computing: Implementation, Management, and Security*. CRC Press.
- Sanjana, T., & Shaveta. G. (2011). Comparative analysis of software development life cycle models, *IJCST* 2(4), pp. 536-539
- Saswade, N., Bharadi, V., & Zanzane, Y. (2016). Virtual machine monitoring in cloud computing. *Procedia Computer Science*, 79, 135-142.
- Scarfone, K., & Mell, P. (2012). *Guide to Intrusion Detection and Prevention Systems (IDPS)* (Draft) (NIST Special Publication 800-94 Rev. 1).
- Sen, J. (2015). Security and privacy issues in cloud computing. In: *Cloud Technology: Concepts, Methodologies, Tools, and Applications* pp. 1585-1630 IGI Global.
- Shah, J. (2015). Implementation and performance analysis of firewall on open switch. Conducted at the Department of Network Architectures and Network Services, Faculty of Informatics Technical University Munich, 29.
- Shah, S., & Mehtre, B. M. (2015). An overview of vulnerability assessment and penetration testing techniques. *Journal of Computer Virology and Hacking Techniques*, 11(1), pp. 27-49.
- Shawish, A., & Salama, M. (2014). Cloud computing: paradigms and technologies. In *Inter-cooperative collective intelligence: Techniques and applications* (pp. 39-67). Springer, Berlin, Heidelberg.

- Shea, R., Wang, F., Wang, H., & Liu, J. (2014). A deep investigation into network performance in virtual machine based cloud environments. In: *INFOCOM, 2014 Proceedings* pp. 1285-1293. IEEE.
- Singh, K., Guntuku, S. C., Thakur, A., & Hota, C. (2014). Big data analytics framework for peer-to-peer botnet detection using random forests. *Inform Sci*, 278, pp. 488–497.
- Singhal, A., & Ou, X. (2017). Security risk analysis of enterprise networks using probabilistic attack graphs. In: *Network Security Metrics* pp. 53-73 Springer, Cham.
- Snapp, S. R., Brentano, J., Dias, G., Goan, T. L., Heberlein, L. T., Ho, C. L., & Levitt, K. N. (2017). DIDS (Distributed Intrusion Detection System)-Motivation, Architecture, and An Early Prototype.
- Sochor, T., & Zuzcak, M. (2014). Study of internet threats and attack methods using Honey pots and Honey nets. In: *International Conference on Computer Networks* pp. 118-127. Springer, Cham.
- Sodhi, B., & Prabhakar T. (2011). A cloud architecture using smart nodes. In: Proceedings of the 2011 IEEE Asia-Pacific Services Computing Conference (APSCC) pp. 116–123, Jeju Island, Korea.
- Souley, B., & Abubakar, H. (2018). A captcha–based intrusion detection model. *Int. J. Softw. Eng. Appl*, 9(1), pp. 29-40.
- Sqalli, H., Al-saeedi, M., Binbeshr F., & Siddiqui M., (2012). UCloud: A simulated Hybrid Cloud for a university environment. In: 2012 IEEE 1st International Conference on Cloud Networking (CLOUDNET) pp. 170–172 IEEE.
- Sridhar, S. (2016). Cloud computing. *An International Journal of Security, Privacy and Trust Management (IJSPTM)*, 4(1), pp. 31-44.
- Stephens, G. D., & Maloof, M. A. (2014). *U.S. Patent No. 8,707,431*. Washington, DC: U.S. Patent and Trademark Office.
- Stergiou, C., Psannis, K. E., Kim, B. G., & Gupta, B. (2018). Secure integration of IoT and cloud computing. *Future Generation Computer Systems*, 78, pp. 964-975.
- Strohmeier, M., Lenders, V., & Martinovic, I. (2014). On the security of the automatic dependent surveillance-broadcast protocol. *IEEE Communications Surveys & Tutorials*, 17(2), pp. 1066-1087.
- Syujak, A. R. (2012). *Deteksi Dan Pencegahan Flooding Data Pada Jaringan Komputer* (Doctoral dissertation, Universitas Muhammadiyah Surakarta).
- Szefer J., & Lee R., (2012). Architectural support for hypervisor-secure virtualization. *SIGARCH Comput.* 40, pp. 437–450.
- Taha, A., & Hadi, A. S. (2019). Anomaly detection methods for categorical data: A review. *ACM Computing Surveys (CSUR)*, 52(2), pp. 38.

- Tao, F., Cheng, Y., Da Xu, L., Zhang, L., & Li, B. H. (2014). CCI-CMfg: Cloud computing and the Internet of Things-based cloud manufacturing service system. *IEEE Transactions on Industrial Informatics*, 10(2), pp. 1435-1442.
- Theoharidou, M., Papanikolaou, N., Pearson, S., & Gritzalis, D. (2013, December). Privacy risk, security, accountability in the cloud. In *Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on* (1, pp. 177-184). IEEE.
- Vance, A., Lowry, P., & Eggett, D., (2015). Increasing accountability through the user interface design artifacts: A new approach to addressing the problem of access-policy violations.
- Vaquero, L., (2011). EduCloud: PaaS versus IaaS cloud usage for an advanced computer science course. *IEEE Transactions on Education*, 54 (4), pp.590–598.
- Varadharajan, V., & Tupakula, U. (2014). Security as a service model for cloud environment. *IEEE Transactions on Network and Service Management*, 11(1), pp. 60-75.
- Veerman, G., & Oprea, R. (2012). Database SQL Injections Detection & Protection. *University van Amsterdam*.
- Velte, T., Velte, J., & Elsenpeter, C. (2010). *Cloud computing: A practical approach* (pp. 44). New York: McGraw-Hill.
- Vieira, K., Schuler, A., Carlos, B., Westphall, C., & Westphall, M. (2010). Intrusion detection for grid and cloud computing, *IEEE Computer Society*, pp. 38-43
- Vieira, K., Schuler, A., Westphall, C., & Westphall, C. (2009). Intrusion detection for grid and cloud computing. *IT Prof Mag*, 4, pp. 38–43.
- Vijayarani, D. S., & Sylviaa, M. M. (2015). Intrusion Detection System: *International Journal of Security, Privacy and Trust Management (IJSPTM)*, 4(1), pp. 31-44.
- Wahlgren, G., Bencherifa, K., & Kowalski, S. (2013). A framework for selecting IT security risk management methods based on ISO27005. In *MIC-CPE 2013: 6th International Conference on Communications, Propagation, and Electronics, Kenitra, Morocco*, pp. 1-3. Academy Publisher.
- Wang, A., Guo, Y., Hao, F., Lakshman, T. V., & Chen, S. (2014, December). Scotch: Elastically scaling up sdn control-plane using vswitch based overlay. In *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies* pp. 403-414. ACM.
- Wang, L., & Jones, R. (2017). Big data analytics for network intrusion detection, a survey. *International Journal of Networks and Communications* 7(1), pp. 24-31.
- Wang, Y., Jodoin, P. M., Porikli, F., Konrad, J., Benezeth, Y., & Ishwar, P. (2014). CDnet 2014: an expanded change detection benchmark dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops* pp. 387-394.

- Weidman, G. (2014). *Penetration testing: a hands-on introduction to hacking*. No Starch Press.
- Wool, A. (2004). A quantitative study of firewall configuration errors. *Computer*, 37(6), pp. 62–67.
- Xia, Y., Liu, Y., Guan, H., Chen, Y., Chen, T., Zang, B., & Chen, H. (2015). Secure outsourcing of virtual appliance. *IEEE Transactions on Cloud Computing*, 5(3), pp. 390-404.
- Xing, T., Xiong, Z., Huang, D., & Medhi, D. (2014). SDN IPS: Enabling software-defined networking based intrusion prevention system in clouds, *10th International Conference on Network and Service Management (CNSM) and Workshop*, pp. 308-311.
- Xiong, Z. (2014). *An SDN-based IPS Development Framework in Cloud Networking Environment* (Doctoral dissertation, Arizona State University).
- Xu, H., Zhou, Y., & Lyu, M. (2016, May). N-version obfuscation. In *Proceedings of the 2nd ACM International Workshop on Cyber-Physical System Security* pp. 22-33. ACM.
- Yesugade, K. D., Avinash, M. S., Satish, N. S., Sandeep, S. C., & Malav, S. (2016). Infrastructure Security Using IDS, IPS, and HoneyPot. *International Engineering Research Journal (IERJ)* 2(3), pp. 851-855.
- Yevdokymenko, M. (2016). An adaptive algorithm for detecting and preventing attacks in telecommunication networks. *Third International Scientific-practical Conference Problems of Infocommunications Science and Technology (PIC S&T)* (2016), pp. 175-177 Kharkiv.
- Zeus (2014) Bot Found Using Amazon’s EC2 as C&C Server Available online: http://www.theregister.co.uk/2009/12/09/amazon_ec2_bot_control_channel/ accessed on 3rd June 2018.
- Zhang, Q., Cheng, L., & Boutaba, R., (2010). Cloud computing: state-of-the-art and research challenges. *Journal of internet services and applications*, 1(1), pp 7-18.
- Zhang, Y., Juels, A., Reiter, M. K., & Ristenpart, T. (2014, November). Cross-tenant side-channel attacks in PaaS clouds. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security* pp. 990-1003. ACM.
- Zhang, Z., Zhang, W., Wang, J., & Chen, X. (2014, April). An Effective Cloud-Based Active Defense System against Malicious Codes. In *Information and Communication Technology-EurAsia Conference* pp. 690-695. Springer, Berlin, Heidelberg.
- Zhao, S., & Medhi, D. (2017). Application-aware network design for Hadoop MapReduce optimization using software-defined networking. *IEEE Transactions on Network and Service Management*, 14(4), pp. 804-816.

- Zissis, D., & Lekkas, D. (2012). Addressing cloud computing security issues. *Future Generation computer systems*, 28(3), 583-592.
- Zou W., Zhang W., Qiang W. and Guofu X. (2013). Design and implementation of a trusted monitoring framework for cloud platforms," *Future Generation Computer System*. vol. 29, 2013, pp. 2092-2102.

APPENDICES

Appendix I: Evaluation Form

Title: A model for Detecting Information Technology Infrastructure Policy Violations in a Cloud

Model Name: PROVIDE

Evaluator's Name: Ruth Anyango Oginga

Date:

Policy Violation Detection Model	Agreed	Disagree	Neutral
PROVIDE Model has the ability to detect and block the threats in real time			
The model solves the issues of false positive			
The set of rules are not manually configured			
PROVIDE Model has the ability to detect and react to encrypted threats			
The model scans the network for the open port and blocks IP of open ports			
The model does not require Human intervention once the attacks have been detected			
The PROVIDE Model monitors the logins and does not allow unauthorized access			

Appendix II: Letter of Introduction



INSTITUTE OF POST GRADUATE STUDIES

Private Bag - 20157
KABARAK, KENYA
E-mail: directorpostgraduate@kabarak.ac.ke

Tel: 0773265999
Fax: 254-51-343012
www.kabarak.ac.ke

12th January, 2018

Ministry of Higher Education Science and Technology,
National Council for Science, Technology & Innovation,
P.O. Box 30623 - 00100,

Dear Sir/Madam,

RE: RESEARCH BY RUTH ANYANGO OGINGA- DGI/M/1272/09/15

The above named is a student at Kabarak University taking PHD Degree in Computer Science. She is carrying out research entitled "*An Architecture for Detecting Information Technology Infrastructure Policy Violations in a Cloud Environment.*"

The information obtained in the course of this research will be used for academic purposes only and will be treated with utmost confidentiality.

Please provide the necessary assistance.

Thank you.

Yours faithfully



Dr. Betty J. Tikoko
DIRECTOR - (POST-GRADUATE STUDIES)

Kabarak University Moral Code

As members of Kabarak University family, we purpose at all times and in all places, to set apart in one's heart, Jesus as Lord. (1 Peter 3:15)

Appendix III: Authorization letter from NACOSTI



NATIONAL COMMISSION FOR SCIENCE, TECHNOLOGY AND INNOVATION

Telephone: +254-20-2213471,
2241349,3310571,2219420
Fax: +254-20-318245,318249
Email: dg@nacosti.go.ke
Website : www.nacosti.go.ke
When replying please quote

NACOSTI, Upper Kabete
Off Waiyaki Way
P O Box 30623-00100
NAIROBI-KENYA

Ref No. **NACOSTI/P/18/92372/22151**

Date: **16th April, 2018**

Ruth Anyango Oginga
Kabarak University
Private Bag - 20157
KABARAK.

RE: RESEARCH AUTHORIZATION

Following your application for authority to carry out research on "*An architecture for detecting information technology infrastructure policy violation in a cloud environment,*" I am pleased to inform you that you have been authorized to undertake research in **Kiambu, Nairobi and Nakuru Counties** for the period ending **12th April, 2019.**

You are advised to report to **the County Commissioner and the County Director of Education, Kiambu, Nairobi and Nakuru Counties** before embarking on the research project.

Kindly note that, as an applicant who has been licensed under the Science, Technology and Innovation Act, 2013 to conduct research in Kenya, you shall deposit a **copy** of the final research report to the Commission within **one year** of completion. The soft copy of the same should be submitted through the Online Research Information System.

**GODFREY P. KALERWA MSc., MBA, MKIM
FOR: DIRECTOR-GENERAL/CEO**

Copy to:

The County Commissioner
Kiambu County.


The County Director of Education
Kiambu County.

Appendix IV: Permit from NACOSTI


THIS IS TO CERTIFY THAT:
MS. RUTH ANYANGO OGINGA
of KABARAK UNIVERSITY, 0-20157
NAKURU, has been permitted to conduct
research in *Kiambu , Nairobi, Nakuru*
Counties


on the topic: **AN ARCHITECTURE FOR
DETECTING INFORMATION TECHNOLOGY
INFRASTRUCTURE POLICY VIOLATION IN
A CLOUD ENVIRONMENT**

for the period ending:
12th April, 2019


.....
**Applicant's
Signature**

Permit No : NACOSTI/P/18/92372/22151
Date Of Issue : 16th April, 2018
Fee Recieved :Ksh 2000




.....
**Director General
National Commission for Science,
Technology & Innovation**

Appendix V: Sample code

Appendix V.1 PROVIDE snort configuration

```
# Custom VRT Rule Packages Snort.conf for Povidе Model

# Setup the network addresses you are protecting
# Internal subnet for povidе network
ipvar HOME_NET 192.168.60.1/24

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

# List of DNS servers on your network
ipvar DNS_SERVERS $HOME_NET

# List of SMTP servers on your network
ipvar SMTP_SERVERS $HOME_NET

# List of web servers on your network
ipvar HTTP_SERVERS $HOME_NET

# List of sql servers on your network
ipvar SQL_SERVERS $HOME_NET

# List of telnet servers on your network
ipvar TELNET_SERVERS $HOME_NET

# List of ssh servers on your network
ipvar SSH_SERVERS $HOME_NET

# List of ftp servers on your network
ipvar FTP_SERVERS $HOME_NET

# List of sip servers on your network
ipvar SIP_SERVERS $HOME_NET

# List of ports you run web servers on
portvar HTTP_PORTS
[80,81,311,383,591,593,901,1220,1414,1741,1830,2301,2381,2809,3037,3128,3702,4343,48
48,5250,6988,7000,7001,7144,7145,7510,7777,7779,8000,8008,8014,8028,8080,8085,8088,
8090,8118,8123,8180,8181,8243,8280,8300,8800,8888,8899,9000,9060,9080,9090,9091,944
3,9999,11371,34443,34444,41080,50002,55555]

# List of ports you want to look for SHELLCODE on.
portvar SHELLCODE_PORTS !80

# List of ports you might see oracle attacks on
portvar ORACLE_PORTS 1024:
```

```

# List of ports you want to look for SSH connections on:
portvar SSH_PORTS 22

# List of ports you run ftp servers on
portvar FTP_PORTS [21,2100,3535]

# List of ports you run SIP servers on
portvar SIP_PORTS [5060,5061,5600]

# List of file data ports for file inspection
portvar FILE_DATA_PORTS [$HTTP_PORTS,110,143]

# List of GTP ports for GTP preprocessor
portvar GTP_PORTS [2123,2152,3386]

# other variables, these should not be modified
ipvar AIM_SERVERS
[64.12.24.0/23,64.12.28.0/23,64.12.161.0/24,64.12.163.0/24,64.12.200.0/24,205.188.3.0/24,205.188.5.0/24,205.188.7.0/24,205.188.9.0/24,205.188.153.0/24,205.188.179.0/24,205.188.248.0/24]

var RULE_PATH /etc/snort/rules
var SO_RULE_PATH /etc/snort/so_rules
var PREPROC_RULE_PATH /etc/snort/preproc_rules

# Set the absolute path appropriately
var WHITE_LIST_PATH /etc/snort/rules
var BLACK_LIST_PATH /etc/snort/rules

# Stop generic decode events:
configdisable_decode_alerts

# Stop Alerts on experimental TCP options
configdisable_tcpopt_experimental_alerts

# Stop Alerts on obsolete TCP options
configdisable_tcpopt_obsolete_alerts

# Stop Alerts on T/TCP alerts
configdisable_tcpopt_tcp_alerts

# Stop Alerts on all other TCPOption type events:
configdisable_tcpopt_alerts

# Stop Alerts on invalid ip options
configdisable_ipopt_alerts

# Alert if value in length field (IP, TCP, UDP) is greater thelength of the packet

```

```

# configurable_decode_oversized_alerts

# Same as above, but drop packet if in Inline mode (requires
enable_decode_oversized_alerts)
# configurable_decode_oversized_drops

# Configure IP / TCP checksum mode
configchecksum_mode: all

# Configure PCRE match limitations
configpcre_match_limit: 3500
configpcre_match_limit_recursion: 1500

config detection: search-method ac-split search-optimize max-pattern-len 20

# Per Packet latency configuration
#config ppm: max-pkt-time 250, \
# fastpath-expensive-packets, \
# pkt-log

# Per Rule latency configuration
#config ppm: max-rule-time 200, \
# threshold 3, \
# suspend-expensive-rules, \
# suspend-timeout 20, \
# rule-log alert

# ConfigurePerf Profiling for debugging

#configprofile_rules: print all, sort avg_ticks
#configprofile_preprocs: print all, sort avg_ticks

configpaf_max: 16000

# path to dynamic preprocessor libraries
dynamicpreprocessor directory /usr/lib64/snort-2.9.12_dynamicpreprocessor/

# path to base preprocessor engine
dynamicengine /usr/lib64/snort-2.9.12_dynamicengine/libsf_engine.so

# path to dynamic rules libraries
dynamicdetection directory /usr/local/lib/snort_dynamicrules

#####
# Step #5: Configure preprocessors

# Does nothing in IDS mode
preprocessor normalize_ip4

```

preprocessor normalize_tcp: ipsecn stream
preprocessor normalize_icmp4
preprocessor normalize_ip6
preprocessor normalize_icmp6

preprocessor frag3_global: max_frags 65536
preprocessor frag3_engine: policy windows detect_anomalies overlap_limit 10
min_fragment_length 100 timeout 180

preprocessor stream5_global: track_tcp yes, \
track_udp yes, \
track_icmp no, \
max_tcp 262144, \
max_udp 131072, \
max_active_responses 2, \
min_response_seconds 5
preprocessor stream5_tcp: log_asymmetric_traffic no, policy windows, \
detect_anomalies, require_3whs 180, \
overlap_limit 10, small_segments 3 bytes 150, timeout 180, \
ports client 21 22 23 25 42 53 79 109 110 111 113 119 135 136 137 139 143 \
161 445 513 514 587 593 691 1433 1521 1741 2100 3306 6070 6665 6666 6667 6668
6669 \
7000 8181 32770 32771 32772 32773 32774 32775 32776 32777 32778 32779, \
ports both 80 81 311 383 443 465 563 591 593 636 901 989 992 993 994 995 1220 1414
1830 2301 2381 2809 3037 3128 3702 4343 4848 5250 6988 7907 7000 7001 7144 7145
7510 7802 7777 7779 \
7801 7900 7901 7902 7903 7904 7905 7906 7908 7909 7910 7911 7912 7913 7914
7915 7916 \
7917 7918 7919 7920 8000 8008 8014 8028 8080 8085 8088 8090 8118 8123 8180
8243 8280 8300 8800 8888 8899 9000 9060 9080 9090 9091 9443 9999 11371 34443 34444
41080 50002 55555
preprocessor stream5_udp: timeout 180

preprocessor http_inspect: global iis_unicode_map unicode.map 1252 compress_depth 65535
decompress_depth 65535
preprocessor http_inspect_server: server default \
http_methods { GET POST PUT SEARCH MKCOL COPY MOVE LOCK UNLOCK
NOTIFY POLL BCOPY BDELETE BMOVE LINK UNLINK OPTIONS HEAD DELETE
TRACE TRACK CONNECT SOURCE SUBSCRIBE UNSUBSCRIBE PROPFIND
PROPPATCH BPROPFIND BPROPPATCH RPC_CONNECT PROXY_SUCCESS
BITS_POST CCM_POST SMS_POST RPC_IN_DATA RPC_OUT_DATA
RPC_ECHO_DATA } \
chunk_length 500000 \
server_flow_depth 0 \
client_flow_depth 0 \
post_depth 65495 \
oversize_dir_length 500 \
max_header_length 750 \
max_headers 100 \
max_spaces 200 \
}

```

small_chunk_length{ 10 5 } \
ports { 80 81 311 383 591 593 901 1220 1414 1741 1830 2301 2381 2809 3037 3128 3702
4343 4848 5250 6988 7000 7001 7144 7145 7510 7777 7779 8000 8008 8014 8028 8080
8085 8088 8090 8118 8123 8180 8181 8243 8280 8300 8800 8888 8899 9000 9060 9080
9090 9091 9443 9999 11371 34443 34444 41080 50002 55555 } \
non_rfc_char{ 0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 } \
enable_cookie \
extended_response_inspection \
inspect_gzip \
normalize_utf \
unlimited_decompress \
normalize_javascript \
apache_whitespace no \
ascii no \
bare_byte no \
directory no \
double_decode no \
iis_backslash no \
iis_delimiter no \
iis_unicode no \
multi_slash no \
utf_8 no \
u_encode yes \
webroot no

```

ONC-RPC normalization and anomaly detection

```

preprocessorrpc_decode: 111 32770 32771 32772 32773 32774 32775 32776 32777 32778
32779 no_alert_multiple_requestsno_alert_large_fragmentsno_alert_incomplete

```

Back Orifice detection.

```

preprocessorbo

```

FTP / Telnet normalization and anomaly detection.

```

preprocessorftp_telnet: global inspection_typestatefulencrypted_traffic no check_encrypted
preprocessorftp_telnet_protocol: telnet \
ayt_attack_thresh 20 \
normalize_ports { 23 } \
detect_anomalies
preprocessorftp_telnet_protocol: ftp server default \
def_max_param_len 100 \
ports { 21 2100 3535 } \
telnet_cmds yes \
ignore_telnet_erase_cmds yes \
ftp_cmds{ ABOR ACCT ADAT ALLO APPE AUTH CCC CDUP } \
ftp_cmds{ CEL CLNT CMD CONF CWD DELETE ENC EPRT } \
ftp_cmds{ EPSV ESTA ESTP FEAT HELP LANG LIST LPRT } \
ftp_cmds{ LPSV MACB MAIL MDTM MIC MKD MLSD MLST } \
ftp_cmds{ MODE NLST NOOP OPTS PASS PASV PBSZ PORT } \
ftp_cmds{ PROT PWD QUIT REIN REST RETR RMD RNFR } \
ftp_cmds{ RNTD SDUP SITE SIZE SMNT STAT STOR STOU } \

```



```

ftp_cmds{ STRU SYST TEST TYPE USER XCUP XCRC XCWD } \
ftp_cmds{ XMAS XMD5 XMKD XPWD XRCP XRMD XRSQ XSEM } \
ftp_cmds{ XSEN XSHA1 XSHA256 } \
alt_max_param_len 0 { ABOR CCC CDUP ESTA FEAT LPSV NOOP PASV PWD QUIT
REIN STOU SYST XCUP XPWD } \
alt_max_param_len 200 { ALLO APPE CMD HELP NLST RETR RNFR STOR STOU
XMKD } \
alt_max_param_len 256 { CWD RNT0 } \
alt_max_param_len 400 { PORT } \
alt_max_param_len 512 { SIZE } \
chk_str_fmt{ ACCT ADAT ALLO APPE AUTH CEL CLNT CMD } \
chk_str_fmt{ CONF CWD DELE ENC EPRT EPSV ESTP HELP } \
chk_str_fmt{ LANG LIST LPRT MACB MAIL MDTM MIC MKD } \
chk_str_fmt{ MLSD MLST MODE NLST OPTS PASS PBSZ PORT } \
chk_str_fmt{ PROT REST RETR RMD RNFR RNT0 SDUP SITE } \
chk_str_fmt{ SIZE SMNT STAT STOR STRU TEST TYPE USER } \
chk_str_fmt{ XCRC XCWD XMAS XMD5 XMKD XRCP XRMD XRSQ } \
chk_str_fmt{ XSEM XSEN XSHA1 XSHA256 } \
cmd_validity ALLO <int [ char R int ] > \
cmd_validity EPSV < [ { char 12 | char A char L char L } ] > \
cmd_validity MACB < string > \
cmd_validity MDTM < [ date nnnnnnnnnnnnnn[.n[n[n]]] ] string > \
cmd_validity MODE < char ASBCZ > \
cmd_validity PORT <host_port> \
cmd_validity PROT < char CSEP > \
cmd_validity STRU < char FRPO [ string ] > \
cmd_validity TYPE < { char AE [ char NTC ] | char I | char L [ number ] } >
preprocessorftp_telnet_protocol: ftp client default \
max_resp_len 256 \
bounce yes \
ignore_telnet_erase_cmds yes \
telnet_cmds yes

```

SMTP normalization and anomaly detection.

```

preprocessorsmtp: ports { 25 465 587 691 } \
inspection_typerstateful \
    b64_decode_depth 0 \
qp_decode_depth 0 \
bitenc_decode_depth 0 \
uu_decode_depth 0 \
log_mailfrom \
log_rcptto \
log_filename \
log_email_hdrs \
normalizecmds \
normalize_cmds{ ATRN AUTH BDAT CHUNKING DATA DEBUG EHLO EMAL ESAM
ESND ESOM ETRN EVFY } \
normalize_cmds{ EXPN HELO HELP IDENT MAIL NOOP ONEX QUEU QUIT RCPT
RSET SAML SEND SOML } \

```

```

normalize_cmds { STARTTLS TICK TIME TURN TURNME VERB VRFY X-ADAT X-
DRCP X-ERCP X-EXCH50 } \
normalize_cmds{ X-EXPS X-LINK2STATE XADR XAUTH XCIR XEXCH50 XGEN
XLICENSE XQUE XSTA XTRN XUSR } \
max_command_line_len 512 \
max_header_line_len 1000 \
max_response_line_len 512 \
alt_max_command_line_len 260 { MAIL } \
alt_max_command_line_len 300 { RCPT } \
alt_max_command_line_len 500 { HELP HELO ETRN EHLO } \
alt_max_command_line_len 255 { EXPN VRFY ATRN SIZE BDAT DEBUG EMAL
ESAM ESND ESOM EVFY IDENT NOOP RSET } \
alt_max_command_line_len 246 { SEND SAML SOML AUTH TURN ETRN DATA RSET
QUIT ONEX QUEU STARTTLS TICK TIME TURNME VERB X-EXPS X-LINK2STATE
XADR XAUTH XCIR XEXCH50 XGEN XLICENSE XQUE XSTA XTRN XUSR } \
valid_cmds{ ATRN AUTH BDAT CHUNKING DATA DEBUG EHLO EMAL ESAM
ESND ESOM ETRN EVFY } \
valid_cmds{ EXPN HELO HELP IDENT MAIL NOOP ONEX QUEU QUIT RCPT RSET
SAML SEND SOML } \
valid_cmds { STARTTLS TICK TIME TURN TURNME VERB VRFY X-ADAT X-DRCP
X-ERCP X-EXCH50 } \
valid_cmds{ X-EXPS X-LINK2STATE XADR XAUTH XCIR XEXCH50 XGEN
XLICENSE XQUE XSTA XTRN XUSR } \
    xlink2state { enabled }

```

Portscan detection.

```
# preprocessorsportscan: proto { all } memcap { 10000000 } sense_level { low }
```

ARP spoof detection

Preprocessors - ARP Spoof Preprocessor

```
# preprocessorarpspoof
```

```
# preprocessorarpspoof_detect_host: 192.168.40.1 f0:0f:00:f0:0f:00
```

SSH anomaly detection.

```
preprocessorssh: server_ports { 22 } \
```

```
autodetect \
```

```
max_client_bytes 19600 \
```

```
max_encrypted_packets 20 \
```

```
max_server_version_len 100 \
```

```
enable_respoverflow enable_ssh1crc32 \
```

```
enable_srverflowenable_protomismatch
```

SMB / DCE-RPC normalization and anomaly detectionpreprocessor dcerpc2: memcap 102400, events [co]

```
preprocessor dcerpc2_server: default, policy WinXP, \
```

```
detect [smb [139,445], tcp 135, udp 135, rpc-over-http-server 593], \
```

```
autodetect [tcp 1025:, udp 1025:, rpc-over-http-server 1025:], \
```

```
smb_max_chain 3, smb_invalid_shares ["C$", "D$", "ADMIN$"]
```

DNS anomaly detection

```
preprocessorndns: ports { 53 } enable_rdata_overflow
```

```
# SSL anomaly detection and traffic bypass
```

```
preprocessorssl: ports { 443 465 563 636 989 992 993 994 995 7801 7802 7900 7901 7902  
7903 7904 7905 7906 7907 7908 7909 7910 7911 7912 7913 7914 7915 7916 7917 7918  
7919 7920 }, trustservers, noinspect_encrypted
```

```
# SDF sensitive data preprocessor.
```

```
preprocessorsensitive_data: alert_threshold 25
```

```
# SIP Session Initiation Protocol preprocessor
```

```
preprocessor sip: max_sessions 40000, \  
ports { 5060 5061 5600 }, \  
methods { invite \  
cancel \  
ack \  
bye \  
register \  
options \  
refer \  
subscribe \  
update \  
join \  
info \  
message \  
notify \  
benotify \  
do \  
qauth \  
sprack \  
publish \  
service \  
unsubscribe \  
prack }, \  
max_uri_len 512, \  
max_call_id_len 80, \  
max_requestName_len 20, \  
max_from_len 256, \  
max_to_len 256, \  
max_via_len 1024, \  
max_contact_len 512, \  
max_content_len 2048
```

```
# IMAP preprocessor
```

```
preprocessorimap: \  
ports { 143 } \  
  b64_decode_depth 0 \  
  qp_decode_depth 0 \  
  bitenc_decode_depth 0 \  
  uu_decode_depth 0
```

```

# POP preprocessor.
preprocessor pop: \
ports { 110 } \
  b64_decode_depth 0 \
qp_decode_depth 0 \
bitenc_decode_depth 0 \
uu_decode_depth 0

# Modbus preprocessor.
preprocessormodbus: ports { 502 }

# DNP3 preprocessor.
preprocessor dnp3: ports { 20000 } \
memcap 262144 \
check_crc

# Reputation preprocessor.
preprocessor reputation: \
memcap 500, \
priority whitelist, \
nested_ip inner, \
whitelist $WHITE_LIST_PATH/white_list.rules, \
blacklist $BLACK_LIST_PATH/black_list.rules

#####
# Step #6: Configure output plugins

# unified2
# Recommended for most installs
# output unified2: filename povidemerged.log, limit 128, nostamp, mpls_event_types,
vlan_event_types
output unified2: filename povidemerged.log, limit 128

# Additional configuration for specific types of installs
# output alert_unified2: filename povidemerged.alert, limit 128, nostamp
# output log_unified2: filename povidemerged.log, limit 128, nostamp

# syslog
outputalert_syslog: LOG_AUTH LOG_ALERT

# pcap
outputlog_tcpdump: povidetcpdump.log

# metadata reference data. do not modify these lines
includeclassification.config
includereference.config

#####
# Step #7: Customize your rule set

```

Appendix V.2 Writing POVIDE Snort Rules

```
#
# NOTE: All categories are enabled in this conf file
#####
# site specific rules
include $RULE_PATH/povide/local.rules
include $RULE_PATH/povide/community.rules
include $RULE_PATH/povide/attack-responses.rules
include $RULE_PATH/povide/icmp.rules
include $RULE_PATH/povide/bad-traffic.rules
include $RULE_PATH/povide/exploit.rules
include $RULE_PATH/povide/urlhaus.rules
include $RULE_PATH/povide/file_identity.rules
include $RULE_PATH/povide/meterpreter.rules
#####
# Step #8: Customize your preprocessor and decoder alerts

# decoder and preprocessor event rules
# include $PREPROC_RULE_PATH/preprocessor.rules
# include $PREPROC_RULE_PATH/decoder.rules
# include $PREPROC_RULE_PATH/sensitive-data.rules
#####
```

Appendix V.3 POVIDE firewall configuration

```
sudo fw route allow in on eth0 out on eth1 to 192.168.171.0/24 from 192.168.3.0/24

route add -net 192.168.3.0/24 gw 192.168.171.4

sudo fw route allow in on enp0s8 out on enp0s9 from 192.168.3.0/24 to 192.168.171.0/24

# Enable masquerading in povide to act as a router to enable traffic from internal subnet to
external network
sudo iptables -t nat -A POSTROUTING -o enp0s9 -j MASQUERADE
sudo iptables -A FORWARD -i enp0s8 -o enp0s9 -m state --state
RELATED,ESTABLISHED -j ACCEPT
sudo iptables -A FORWARD -i enp0s9 -o enp0s8 -j ACCEPT
```

Appendix V.4 POVIDE detects and blocks attacks traffic

```
firewall-cmd --zone=public --add-masquerade --permanent

#-----
# LOCAL RULES
#-----
#alert icmp any any -> $HOME_NET any (msg:"ICMP test"; sid:10000001; rev:001;)
alert icmp any any<> 192.168.171.8 any (msg: "Povide Packet found"; sid:10000001; )
alert tcp any any -> any 21 (msg: "Povide FTP Packet found"; sid:10000002; )
alert tcp any any -> any 22 (msg: "Povide SSH Packet found"; sid:10000003; )
alert tcp any any<> any 80 (msg: "Povide HTTP Packet found"; sid:10000004; )
#alert tcp any any<> any 443 (msg:"Povide possible attack traffic detected no actions taken";
classtype:trojan-activity; sid:100000023; )
```

```
alerttcp any any<> any 443 (msg:"possible attack traffic detected and blocked";
classtype:trojan-activity; react: block, msg; sid:10000005; )
```

```
#!/bin/bash
```

Appendix V.5 POVIDE scans the network and blocks open ports

```
ipresults="$(povide -sT 192.168.60.0/24 --exclude 192.168.60.60 |
awk '/^Povide scan report/{cHost=$5;}
/open/ { split($1,a,"/"); result[cHost][a[1]]=""}
END {
for (i in result) {
printf i;
for (j in result[i])
printf ",%s", j ;
print ""} }' |
sed -e 's/,^t/' | awk '{print $1}')"
if [ -n "$ipresults" ]; then
ufw deny from $ipresults to any
ufw disable
ufw enable
echo "Povide Blocked IP: $ipresults and access denied by firewall" >> blocked.txt
fi
ufw allow from 192.168.60.91 to any
```