Master of Science in Computer Science Engineering
October 2017

Evaluation of Intrusion Detection Systems under Denial of Service Attack in virtual Environment

Comparative study of Snort, Suricata and OSSEC

Venkatesh Nagadevara

Faculty of Computing
Blekinge Institute of Technology
SE-371 79 Karlskrona Sweden

This thesis is submitted to the Faculty of Computing at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science in Computer Science Engineering. The thesis is equivalent to 20 weeks of full time studies.

Contact Information:
Author(s):
Venkatesh Nagadevara
E-mail: vena15@student.bth.se

University advisor:
Emiliano Casalicchio
Department of Computer Science and Engineering

Faculty of Computing
Blekinge Institute of Technology
SE-371 79 Karlskrona, Sweden

Internet : www.bth.se
Phone : +46 455 38 50 00
Fax : +46 455 38 50 57

ABSTRACT

Context. The intrusion detection systems are being widely used for detecting the malicious traffic in many industries and they use a variety of technologies. Each IDs had different architecture and are deployed for detecting malicious activity. Intrusion detection system has a different set of rules which can defined based on requirement. Therefore, choosing intrusion detection system for and the appropriate environment is not an easy task.

Objectives. The goal of this research is to evaluate three most used open source intrusion detection systems in terms of performance. And we give details about different types of attacks that can be detected using intrusion detection system. The tools that we select are Snort, Suricata, OSSEC.

Methods. The experiment is conducted using TCP, SCAN, ICMP, FTP attack. Each experiment was run in different traffic rates under normal and malicious traffics all rule are active. All these tests are conducted in a virtual environment.

Results. We can calculate the performance of IDS by using CPU usage, memory usage, packet loss and a number of alerts generated. These results are calculated for both normal and malicious traffic.

Conclusions. We conclude that results vary in different IDS for different traffic rates. Specially snort showed better performance in alerts identification and OSSEC in the performance of IDS. These results indicated that alerts are low when the traffic rates high are which indicates this is due to the packet loss. Overall OSSEC provides better performance. And Snort provides better performance and accuracy for alert detection.

Keywords: Intrusion Detection System, Snort, OSSEC, Suricata, Network Traffic.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 INTRODUCTION

Cloud computing is one most growing industries in the field of computer science. This field mainly deals with IT services such as shared networks and computer resources (servers, storage, applications, data, etc.) for the users. This cloud mainly contains data that is sensitive and valuable to hackers. Due to this, there is an increase of attacks on the cloud. To prevent this security intrusion there are two kinds of systems intrusion detection system and intrusion detection system [1]. This thesis is mainly focused on intrusion detection systems.

Intrusion detection system is a process of detecting any malicious activity in the network. It mainly deals with unauthorized access to the system. IDS analyze the incoming traffic and observes for any malicious activity in the data or files that are incoming. Intrusion detection system cannot prevent all attacks it can only be used for detection and they are more effective if we use it as a combination of intrusion prevention systems. The intrusion detection systems are as follows.

- *Host-based intrusion detection system*: It is present on the system such as local system or server. It basically analyzes the file's data that are incoming and tries to verify with current rules and informs to the administrator if any malicious activity occurs. It must be present in every host system[2]. Due to this may decrease the system performance [3].

- *Network based intrusion detection system*: This is present on the network and can monitor attack on multiple hosts at the same time without effecting the host systems performance. It is difficult to manage when compared to host-based intrusion detection system [4].

- *Hypervisor based intrusion detection system*: It usually indicates a level of communication between the virtual machines that are present in hypervisor layer. It helps to analyze the actions of user. And the communication between virtual machines, network and hypervisor based virtual network.

- *Distributed intrusion detection system*: In distributed type, we have the host based intrusion detection systems and network based intrusion system are installed on to network which constantly analyzes the incoming network for any malicious behavior. They have two major components detection component and correlation manager. The detection component analyzes the behavior system and report it to the correlation manager which gathers the information from multiple intrusion detection system and generates an alert if any malicious activity takes place.

- *Signature based intrusion detection*: Signature based detection is done by comparing the suspicious activity with the signature based rules. Generally, these rules contain patterns of attacks that are previously defined. It cannot detect unknown attacks that are not defined [5].

- *Anomaly based detection*: Anomaly based detection in which the current user activity is measured against the previous logs of the user. It may not be so reliable because irregular behavior of user it may produce more false alarms. It may also require huge data to get the results this will increase the time and memory usage [6].

- *Hybrid detection technique*: Hybrid type detection technique uses both anomaly and signature detection technique. It is more effective because it can detect both known attacks by rules set and unknown attacks by observing attack patterns.

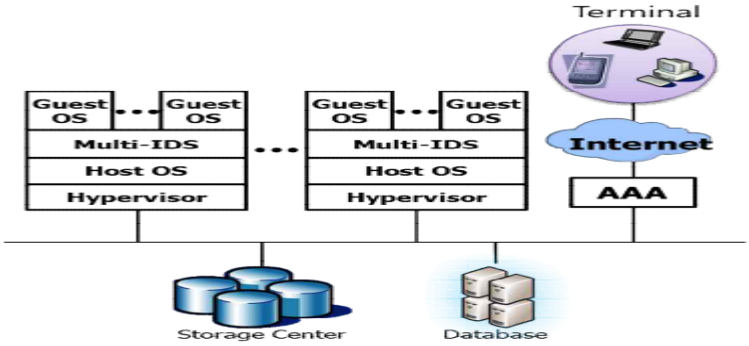

Figure 1-1: Distributed Intrusion Detection System [3].

1.1 Motivation

The intrusion detection system tools are widely used in the industry for detection of any type of intrusion. There are many types of open source intrusion detection systems that are present in the market among them some are Snort, Suricata, Bro, OSSEC, Security Onion, OpenWIPS-NG, etc.

- *Snort:* Snort is the most often used open source network intrusion detection system. In this the attacks are detected by the set of rules. This usually analyzes all packets that are incoming.

- *Bro:* Bro is also a network based open intrusion detection system. We can write rules in separate scripting language. It is able conduct network monitoring and handle high network performance.

- *Security Onion:* Security Onion is also open source Linux based intrusion detection system. It can be used as a sensor for detecting malicious activity in virtual machines, lan's and subnet. It uses other intrusion detection systems like suricata, OSSEC, bro.

- *OSSEC*: OSSEC is an open source host based intrusion detection system. It performs log check, files integrity analysis that are incoming, root kit detection and real time alert generation.

- *Open WIPS-NG*: Open WIPS-NG is an open source intrusion detection and prevention system. There are already many services that are built into aircrack-ng that are useful for detecting and preventing intrusions.

- *Suricata*: Suricata is also an open source network intrusion detection system that is like snort. Which detects the intrusion with help of ruleset that are predefined in its rules. The main difference is multi-threading.

Many researches have used different approaches to measure the performance of intrusion detection system. However different types of alerts are generated for different rules. So, this thesis measures the performance between bro, suricata, OSSEC.

1.2 Problem statement

The intrusion detection system is difficult to implement because they are specifically designed to run on specific system and environments. Thus, the techniques and rules the present in latest version of intrusion detection system. Moreover, in this study we are evaluating three different intrusion detection system under attack and normal traffic to measure the performance individually and compare with each other.

1.3 Aims and Objectives

The aims and objectives this thesis are formulated to gain a deeper knowledge on IDS present in cloud system and their working mechanisms. Aim and Objectives are described below:

1.3.1 Aim

This thesis study is to understand the state of the art related to Intrusion detection systems for cloud computing.

1.3.2 Objectives

1. To obtain knowledge about IDS techniques by focusing on cloud systems security issues and protection.
2. To obtain knowledge about the cloud computing and wide ranges of cyber-attack.
3. To determine which IDS in cloud system that can be used against a certain typical attack.
4. To understand how IDS performance can be measured.
5. To compare the performance of IDS systems for selected metrics.

1.4 Research Questions

The following are the research questions set to achieve the aim and objectives of this thesis:

RQ1: What are the state of the art solutions for Intrusion Detection System in cloud computing?

RQ2: What are the main types of cloud attacks that can be detected using Intrusion Detection System?

RQ3: What is the performance of different Intrusion Detection System?

1.5 Structure of Thesis

Thesis is structured in the following manner:

- Chapter 1: This chapter describes about the introduction, objective, motivation, and problem statement
- Chapter 2: Background. This chapter describes about the background. Intrusion detection systems their rules and their architecture
- Chapter 3: Literature review. This section reviews the literature that are useful to this research.
- Chapter 4: proposed work. This contains the propped architecture, different types of attacks that can be detected by intrusion detection system and tools that are required for experimentation.
- Chapter 5: Experimentation and results. This contains experiment setup and results that are obtained by running experiment.
- Chapter 6: Conclusion. This summary of the experiment conducted and limitation

2 BACKGROUND

This chapters explains the three-intrusion detection that are used for detection and the traffic generator. And the frame work of common intrusion detection system. The intrusion detection systems that are used are Suricata, OSSEC and Snort.

2.1 Framework for common Intrusion detection system

There are four main parts in intrusion detection system. They are shown as following in figure incident storage, analyzers, response unit and sensors.

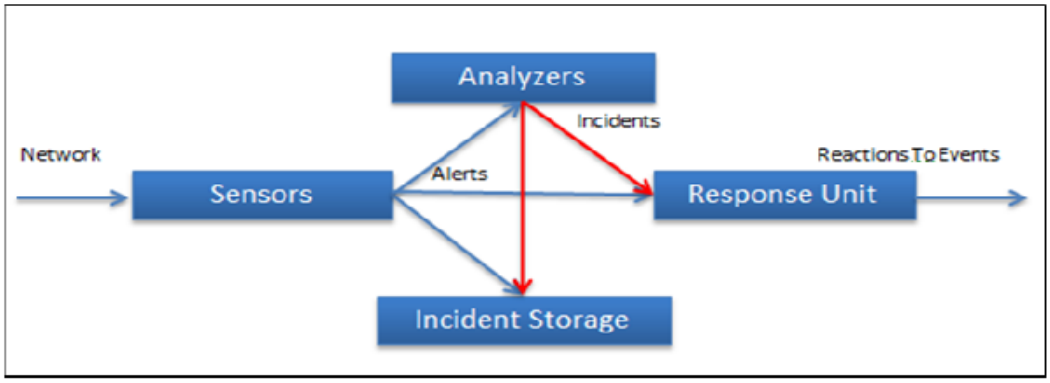

Figure 2-1: Components of CIDF [7].

2.2 Suricata

Suricata is developed by Open Information Security Foundation (OISF). It is developed to be an intrusion detection and prevention system. It was built as an alternative for snort. Suricata libraries and engine can be used under GPLv2. It is a both intrusion detection and prevention system that uses ruleset that written and imported to detect any malicious activity in the incoming network traffic. It is compatible with existing network components. Multi-threading is main advantage in suricata.

2.2.1 Architecture

This is a rough architecture on suricata based on snort. This can defined using pipeline and suricata model. The pipeline uses libpcap for capture and storing packets that are incoming traffic.

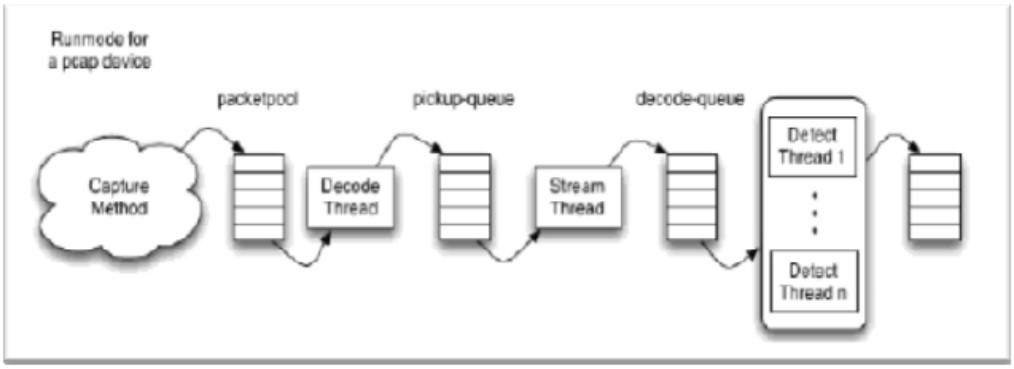

Figure 2-2: Suricata Architecture [7].

The whole process of pipeline is initiated when the suricata is in run mode. Then the thread is initialized which initializes the packet processing between the modules and queues. The threads are in runnable state when all modules are in run state.

- Capture mode: We use pcap to capture the incoming data we can initialize it by using eth0 command based on what network you are using. When it is initiated it will start capturing all packets and the send it to decoder. Libpacap is limited to process individual data packets.

- Decoder module: Once the packets are received to suricata by pcap. Suricata starts process of decoding by moving through preprocessors.

- Detection module: Then the packets are reached they are already decoded. The preprocessor observes for any types of malicious behavior by comparing the traffic with signature based rules for any patterns of attack.

2.2.2 Suricata Rule Structure

The signature detection plays a big role in suricata. The suricata uses common ruleset and most familiar emerging threats, emerging threats pro and Sourcefire's VRT. This an example how a rule looks like below.

Table 2-1: Suricata Rule Structure [7].

<Action><Protocol><Source IP><Direction><Source Port><Destination IP>< Destination Port > (<Rule Options>) |< - Action - >|<----------Header----------> >|< ----Options---- >|

The main portions of rule are as follows [8]:

I. Action order

Signature have many types of qualities. Among them important one is action property. There are four possibilities when a signature matches those are as follows:

- Pass: When an signature is passes then suricata gets to the end of the rules only for the analyzing packet.
- Drop: It basically deals with intrusion prevention system when program contains signature as drop it stops scanning and does not process any further. Suricata will count it as packet loss alert.

- Reject: Both sender and receiver will receive a reject packet when signature is reject. There are two types of responses that are if it tcp then it will reset packet. I f it other any protocol then the protocol is icmp-error packet.
- Alert: When signature is alert it will be treated as normal attack then the intrusion prevention system will either use drop or reject to protect. The user and intrusion detect can act to alert.

```
drop tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"ET
TROJAN Likely Bot
Nick in IRC (USA +..)"; flow:established,to_server;
flowbits:isset,is_proto_irc; content:"NICK "; pcre:"/NICK
.*USA.*[0-9]{3,}/i"; classtype:trojan-activity;
reference:url,doc.emergingthreats.net/2008124;
reference:url,www.emergingthreats.net/cgi-
bin/cvsweb.cgi/sigs/VIRUS/TROJAN_IRC_Bots;
sid:2008124; rev:2;)
```

Figure 2-3: Suricata rule example [8].

II. Protocol

Signature tells which protocol it contain. You have four options udp, tcp, icmp and ip. Suricata adds an few more protocols dns, ftp,http,tls and smb. These are called application layer protocols which are used to determine traffic that is incoming.

III. Source and destination

We can give ip address In yaml file you can give address to home.net and external.net.

IV. Ports

The traffic is transmitted through port you can give a specific port to scan and other advanced option.

V. Direction

This gives the direction that a packet has to be matched with the signature. Nearly every signature that we specify has arrow pointing right.
Example:

alert tcp 1.2.3.4 1024 - > 5.6.7.8 80

VI. Rule options

It binds a rule to a rule binding system. There are many types some of them are described below.

- Msg (message) it gives details about the signature and possible alert.
- Rev (revision) refers the version of signature.

Sid (signature ID) It specifies the identy of the signature and group id.

2.3 Snort

Snort is one of the well-known intrusion detection systems. It detects the malicious using signature based detection. It is an open source intrusion detection system. It has the capability to monitor IP based network traffic in real time. This is developed by Martin Roesch who is a founder and chief technology officer at Sourcefire. The snort is sold by the Sourcefire to the companies by the source fire with their hardware.

It is mainly used to either actively or passively detect the malicious attacks like buffer overflow, port scans, DOS, DDOS, web application attack and other attacks. It is mainly used for intrusion prevention. Snort has a wide range of uses and community support.

2.3.1 Snort Architecture

Snort architecture is as shown in the below figure. It contains six components.

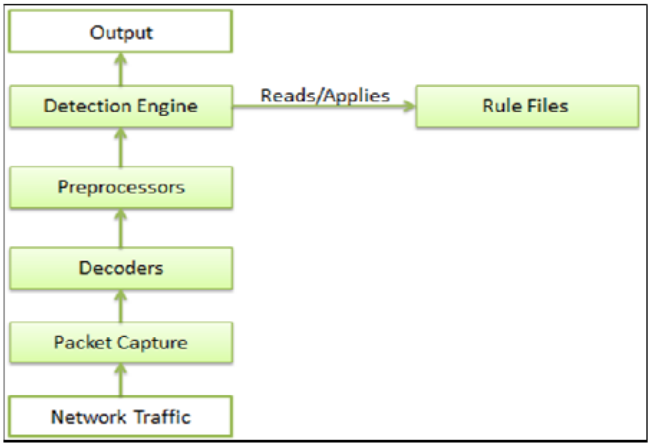

Figure 2-4: Snort Architecture.

- *Decoders:* Decoders stores the packets in the form of data structures and Then uses protocols to decode the packets to get useful information such as port id, destination address and other. Snort will give an error alert if it has unidentified headers, or TCP length is more than usual.

- *Preprocessors:* This is a filter that is used to filter things that are next examined by detection engine. This can also be used to detect malicious TCP, UPD port connections.

- *Rules Files:* This files are text files that contains rules that are used for malicious activity detection. This files contains syntax that includes protocols, addresses, plugins, file locations and other. This rules files can be updated and we can also define custom rules.

- *Detection Plug-ins:* This are used to detect the pattern using the rules that are evaluated.

- *Detection Engine:* These uses detection plug-ins. This uses rules that are previously loaded into the memory when it is started.

- *Output Plug-ins:* This allows the user to access the notification and results via alerts and logs that can be accessed by user. It can be accessed by files, logs or databases.

2.3.2 Snort Rule Structure

The rule structure of snort follows simple protocols which makes rule designers task easy. The structure is mainly equipped with the rule headers and rule options. The rule headers contain the required information which is common to every rule. The rule options are designed to refine the filter capabilities of snort. The rule options are optional and vary for each rule. The basic rule structure is shown below.

<Action><protocol><Source IP><Direction><Source port><Destination port> (<Rule Options>)

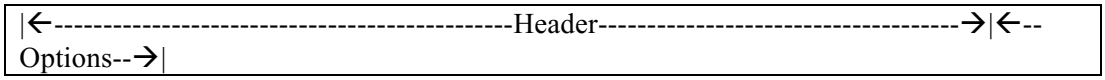

Figure 2-5: Snort Rule Structure

The Rule header has five main components, they are:

- Action: This tells snort what action to perform if there is a match found for the rule triggered in snort.conf file. The general action for the snort is logging the information in alert file. There are eight action options in snort. They are alert, log, pass, activate, dynamic, drop, reject and sdrop. The last three options mentioned in the list are enabled only when snort is running in inline mode.

- Protocol: the protocols like TCP, UDP, ICMP or IP are specified.

- IP Source and IP destination: The IP address of the source or destination is defined. The general syntax for defining IP source and IP destination are CIDR (Classless Inter-Domain Routing) block, a range of IP addresses, or the string "any" for all possible IP address.

- Port Source and Port Destination: The port number for the source or destination is defined. The general syntax for defining port number is by defining the specified static port number or the string "any" for all possible port numbers.

- Direction: This defines the orientation of the rule applied. The operator used was "→"

The rule options are built with five components. They are

- Rule options: These provide the detail of matching parameters to bind a rule to a rule identification system.

- General: The information about reference information, specific log messages/alerts, rule identification is provided by these options.

- Payload: To examine the data contained in the network packets such as packet header, these options are used.

- Non-Payload: To provide the matching specifications against packet headers these options are considered. The port numbers and IP address are excluded.

- Post-Detection: These options include time-to- live values, specific IP options and fragment offsets.

2.4 OSSEC

Ossec is a host based intrusion detection system. Which is used to monitor logs and security information management and event management. They are combine a effective intrusion detection system. It helps the user to detect unauthorized usage of files and alerts to user. It lets you configure alerts for certain attacks. It also enables email enable alerts because of features like smtp, sms and syslog. The main key features are log monitoring, root kit detection and active response. The action response helps to act on alerts immediately when generated.

2.4.1 OSSEC Architecture

The main components of OSSEC architecture are in the below figure

Figure 2-6: OSSEC architecture [9].

- Server: The server is the main processing unit OSSEC deployment. The server stores files, log history, log events. Rules and decoders that are essential to intrusion detection system to function are present in server. So it is easy to access to required data even for large agents.

- Agents: Agent is a main part of intrusion detection system it majorly consists of small or large programs that are installed on system to observe the incoming data. This will collect the data that it received and sends it to further analysis Agentless is a service that allows that we cannot install. It helps in checking the activity in server. It can be used to monitor firewall, applications and operating system.

- VMware: Virtual machine is helpful to install operating system on it which OSSEC can use to install agents on to them. Which helps to determine actions like when a guest has logged in and logged out. It helps in maintain center of internet security and generates alerts if a malicious activity or setting are in insecure state. But this may cause support issues.

- Firewalls, switches and routers: Ossec analyzes syslogs from firewall, routers and switches. There are many types of routers among them some are cisco routers, checkpoint, netscreen etc.

2.4.2 OSSEC Rule Structure

Ossec already has the default rules that are installed during installation process. Which helps it to detect a wide range of attacks. If the number of rules increase then false positive increases.

```
<rule id="502" level="3">
  <if_sid>500</if_sid>
  <options>alert_by_email</options>
  <match>Ossec started</match>
  <description>Ossec server started.</description>
  <info type="link">http://ossec.net/wiki/Rule:205</info>
  <info type="cve">2009-1002</info>
  <info type="osvdb"> 61509</info>
  <info type="text">Internal Why we are running this run in our company</info>
  <info>Type text is the default</info>
</rule>
```

Figure 2-7: OSSEC rule example [10]

SSH rules: The ae three ssh decoder rules show in relation with other. The rules are as follows

- <group>tag
  Groupid creates logical grouping of rules which are being utilized in file configuration.

- <rule id>tag This contains a lot of range of rules default from 0000-99999 and custom rules from 100000-119999. Level is also specified in rule id tag. This also can has disable option in which we can disable certain rules to observe the p the other rules.

- <decode as>tag
  logs are decoded by ssh decoder. This logs are within scope of the rules of OSSEC.

- <match>tag:

It is a text matching if any pattern matches with the rules then particular rule is triggered.

2.5 Summary

The below table show the features and compare with other intrusion detection system whether features like ipv4 are there or not.

Table 2-2: Comparison of Three IDS [7]

Features	Ossec 2.8.3	Suricata 3.2.1	Snort 2.9.9.0
IPv6	NO	YES	NO
Multi-threading	YES	YES	NO
GPU	YES	YES	SOME
Automatic protocol detection	YES	YES	YES
IPv4	YES	YES	YES

Global variable	YES	YES	NO
Windows support	YES	YES	YES
Ubuntu support	YES	YES	YES
GeoIP	YES	YES	YES
Open BSD	YES	YES	YES
Http accessing	YES	YES	YES
Advanced http	YES	YES	YES
SMB logging	NO	YES	NO
Http blocklist lookups	YES	YES	YES
Free of cost	YES	YES	YES

3 METHODOLOGY

3.1 Introduction

There are five research methods [][] in computer science and they are experimentation, conducting survey, conducting interview, literature review and simulation. In this thesis the first and second research questions are answered by literature review. Because according to the kitchenham [19] "literature review is the best method for identifying, evaluating and interpreting all available research relevant to a particular research question, or topic area, or a phenomenon of interest." The research question 1 and 2 are answered by doing the literature review.

The main reason to understand the state of art of IDS and determine the attacks that can be identified using IDS. The following are the some of the techniques and the methods described in the paper. The literature review is done based on refining the literature from databases that contain research papers. The papers are sorted by using key word that are related to research following are some of the papers that related to research. The literature review also contains the comparison of some intrusion detection systems that are available.

- Planning: This discusses about need of literature review and protocols used.
- Conducting: In this we can determine the literature that can be helpful to select procedure for thesis.

The description of planning and conducting are as below

3.2 Planning literature review

3.2.1 The need for literature review

The main objective to perform the literature review for this study is to how to measure the performance for intrusion detection systems and what are different attacks that can be detected by using selected intrusion detection system. Next step based on the data that is obtained we use the IDS and method to evaluate the experiment.

3.2.2 Research question

These are research question which are answered through conducting literature review are:

- **RQ1:** What are the state of the art solutions for Intrusion Detection System in cloud computing?
  How: By using the literature review we found out the different intrusion detection systems and their use in cloud computing to prevent intrusions.

- **RQ2:** What are the main types of cloud attacks that can be detected using Intrusion Detection System?
  How: The literature review will help to determine many attacks that can be detected by selected intrusion detection systems.

- **RQ3:** What is the performance of different Intrusion Detection System?
  How: The literature review helps finding out the metrics that can be used to evaluate the intrusion detection systems and compare them.

3.2.3 Keyword selection

The keyword is selected using the the guidelines used in []. The criteria used to select the keywords in this is are based on Population Intervention Comparison Outcomes (PICO).

- Population: This discuses about the main area of research that is cloud computing.
- Intervention: In this Intrusion detection systems are used to detect the attacks in this research so it is selected and keyword.
- Comparison: The comparison is done between different intrusion detection systems so we select evaluate intrusion detection system.
- Outcome: In the comparison of intrusion detection the most impotent is accuracy so it is selected as keyword.
- Context: The comparison of intrusion is in academic environment so the academic is context.

The search string in IEEE is as follows ((“cloud computing” OR virtualization) AND (Intrusion detection system) AND (performance evaluation). The date is adjusted between 2010-2017. The papers are selected by relating to the research questions. 14 are selected from them.

3.2.4 Inclusion and Exclusion criteria

Inclusion criteria

- Studies based on comparison of intrusion detection systems against attack.
- Performance evaluation in proposed solution.
- Papers published between 2010-2017.
- Articles published in English language.

Exclusion criteria

- The studies that are not related to Intrusion detection systems.
- Studies that did not use performance metrics.

3.3 Literature review

The following are the some of the useful methods and papers that are useful to carry our experiment.

This papers compares two intrusion detection and prevention system under various attacks in different operating systems. The intrusion detection systems mentioned in the paper are snort and suricata. This study gives a more detailed comparison of performance of intrusion detection systems under attacks in different environments and the benchmark taken is CPU usage and RAM usage [1].This experiment is conducted as follows it contains server and client. The clients generate traffic and sends it to server for analysis. This experiment contains 50tests which each last for one minute. In each each experiment, we capture network traffic using tcpdump and save captured packets in pcap file. The CPU usage and memory usage are read after every experimentation module [1].

In the first part of experiment are results are as follows [1]:

- The difference between the CPU usage between snort and suricata in windows and Linux are negligible.
- The memory usage between snort and suricata in windows and Linux are no different.

- There is no difference between number of packets that are dropped in snort and suricata respectively in windows and Linux systems.

The experimentation concludes that the Linux uses more resources that windows this concludes that the performance not only depends on intrusion detection system but also operation system.

This papers discusses the performance of three popular intrusion detection systems such as snort, suricata and bro. The performance is measured under various attacks. The metrics that are considered for comparison are CPU utilization, packet loss, memory utilization and number of alerts that are generated [7].The framework mainly contains of three components network generation, intrusion detection and alet generation. The author used four network generators such as ostinato, nmpap, High orbit ion canon and low orbit ion canon for generation of attacks. In this project the rules are are selected from the existing threats from open source community project. The rules of snort and suricata are similar. They particularly selected eight kinds of attacks [7].
Results: Intrusion detection systems uses centos V5.0. They are designed in a different architecture. They use different signature rules for detecting the intrusions. They evaluated the intrusion detection system in normal traffic and malicious traffic. And at different traffic rates. And measure the metrics like memory usage, packets lost, cpu usage and alerts generated [7].

- Their experiments results were the low packet loss and low cpu usage in TCP traffic.

- High speed packet has significant effect on number of alerts generated, cpu usage, memory usage and packet loss.
- Three intrusion detection system uses the resources and acts differently for each type of attack.
- Three intrusion detection systems also have different levels of packet loss and alerts generation.

- Finally, different set of rules can generate different number of attacks.

This paper focuses on the comparison of three open source intrusion detection system that are snort, suricata and bro. The thesis consists of advantages and disadvantages of each intrusion detection system. The performance of each intrusion detection system on one gigabits with several experiments conducting. Snort has been an open source intrusion detection system that is becoming popular in industry. Suricata is a new open source intrusion detection system that is like snort but has additional features like multi-threading. Bro is some different than other intrusion detection systems it provides comprehensive analysis on network traffic [11]. The experiment conducted with all three with their default rule set. The intrusion detection is set that it can only handle 100Mbits/s. After optimization using PF_RING network socket, performance is increased at least ten times. Transmission of packets at speed of 1000 Mbit/s are handled without any packet loss [11].

This paper discuses about two popular intrusion detection systems namely snort and suricata. Which are analyzed under high speed networks. The experiment is conducted in three platforms ESXi (virtual machine), Linux and open BSD to measure performance under traffic [12].

- The intrusion detection systems such as snort and suricata are installed on the three operating systems.
- All the intrusion detection systems are tested on different traffics and packet sizes respectively.

- The speed are about 250Mbits/s, 500 Mbits/s, 750 Mbits/s, 1Gbits/s,1.5 Gbits/s, 2 Gbit/s and 2.5 Gbi/s
- In all scenarios, the snort and suricata have same set of rules to run and monitor the incoming traffic.

This paper analyzes data for mobile networks. The data sets of the mobile networks are listed as below [13]. RTT-IM and RTT-5M million packets taken from a link.
They used rules of VRT via the Snort2Bro software. The parameters taken are number of alerts and analysis time.Understanding IPS and IDS: Using IPS and IDS together for defense in depth
This paper mainly focuses on intrusion detection system and prevention system together. This paper mainly focuses on architectural view of intrusion detection and prevention system. It describes how the IDS can be used to protect against malicious attacks. Distributed denial service attack on cloud : detection and prevention[17]
This paper consists of detection and prevention of DDOS attack in detail. The objective was to present a tool that could help to detect the DDOS attack. The features and how the tool is deployed is explained. It contains two experiments on with the IDS is activated the attack happens and another it's not active. The CPU usage is monitored in the both cases and it is observed in the experiment how snort can be helpful to improve the performance under DDOS attack. The DDOS attack is a serious threat because it almost consumes the CPU usage.
Defense of DDOS attack for cloud computing [18]
This paper uses a track back system to detect and prevent DOS and DDOS attacks in cloud based. To the SOA packet a tag is added by which we can determine the attacker. In this paper spoofed IP is not considered only IP is considered so it might not so effective attacker uses spoofed IP.
Rate based intrusion prevention system against DDOS [19]
This paper discuses rate based approach to prevent DDOS attack. This evaluates system background traffic, attack traffic with four DDOs attacks. The performance metrics that are considered in snort are bandwidth, reliability, latency, CPU and memory usage. The results shows that the IDS is efficient to detect small scale DDOS attacks. IT is also observed that IDS drops packets when the traffic rate is high.
Securing cloud from DDOS attack using IDS in virtual system. [20]
In this author proposed IT virtualization strategy to protect them against DOS and DDOS attacks. In this process an Intrusion detection like snort is installed to monitor incoming traffic and outgoing traffic. When there is pike in graph we determine that sender side traffic is observed. If acknowledgements are not received, then IDS request the honeypot to acquire the IP address. If the reply is not received, then it is considered as an DOS or DDOOS attack and then the sender IP is blocked.

4 EXPERIMENT

The main elements of architecture are incoming traffic, intrusion detection system software and rule set for alerts generation. The incoming traffic is generated by traffic generator. It uses tcpdump to send to each intrusion detection system. The second part of the architecture is rule set to detect intrusions. The third part is tools Suricata, Snort and OSSEC. The intrusion detection systems are installed with basic configurations.

4.1 Types Of Attacks that can be Detected Using Intrusion Detection Systems.

In this thesis, we have seen may intrusions that can compromise systems security. The intrusion detection system determines these intrusions with the rules set. Some of the attacks are as follows.

- *Bad traffic:*
  This attack sends illegal packets that are sent. Which uses TCP and UDP port 0 and they may also use SYN packets forecast address more than once.

- *Scanning attack:*
  Scanning attack can be used to detect system being attacked. Using scanning attack, the hacker can know information like topology information, network traffic that are accepted in firewall, operating system, os kernel, network kernel and software that are running and their versions. By gathering this information, the attacker launches a attack on particular attack. These are achieved by using stealth scan SYN attack. This stealth because it does does not establish tcp connection. This is also called as half open scanning. The attacker sends SYN packets to get information about port, If the attacker receives any SYN/ACK then it means port is open and if not assumes that port is closed.

- *Denial of service attack:*
  The two main types of denial of service namely flaw exploitation and flooding. In flooding attack, we launch attack by just using ping command. This will see the host packets if the attacker has higher bandwidth than host attack completed fast. Another example is SYN packets that are sent to make half connection contains forged address. After making connection with the host the host sends an acknowledgement and keeps waiting for return acknowledgement and wastes resources on it [6].

- *Penetration attack:*
  Penetration attack is an effective attack which will give the attacker all the details such as resources, data, applications etc., attacker uses the exploits in software to launch attack. The penetration attack will give attacker root access. They decide to launch an attack on that system or use that system to launch an attack on another system.

- *ICMP attack*
  ICMP is used internet protocol layer to send messages to host. The authentication process is not present in ICMP that why in denial of service or attackers can catch packets [14]. There are few types of ICMP attacks

ICMP DOS Attack: This attack uses The ICMP "Time exceeded" and it can also use "destination unreachable" messages. This will cause the connection of host to disconnected state. The communication between server and host will be disconnected.
Ping of death: the attackers sends an ICMP echo that is greater than the IP protocol packet size. Since the received packets are more than it can handle it crashes and sent to reboot state.
ICMP nuke attack: In this attack the attacker send packets that os can handle and crashes it with ICMP flood attack. The traffic becomes so high it stops responding to normal traffic [14].

- *TCP attack*
  When we establish connection with ip by hand shaking then full duplex communication between applications occur, there are many security issues with TCP connection.
  TCP SYN: In this process the attacker send a packet to address with ACK header then it establishes connection with server then send packet and receives acknowledgement but never sends back response. Then server send messages for response and attacker floods with TCP SYN packets and wastes the server memory and sends it to non-responding state
  TCP sequence number attack: The goal of this attack is to take control of the attack system receiving data from server. Every time TCP connection is established the messages are sent with a sequence number. If the attacker can obtain this sequence number, he can be between the attacked system and server. He can receive the data from sever as a trusted client that was intended to send to the attacked system.

- *ARP flooding*
  The Arp table contains huge sets of data and consumes the resources of the computer. So, the size of the table is constant attacker sends a large table data set to ip that it cannot handle and the system uses it resources for resolving it. Meanwhile the attacker sends many sets of table to random ip address the system stops responding due to overload [15].

- *UDP*
  The UDP is a simple transmission method that is non-reliable, data authenticity is not checked. Thus, it is not a reliable service,
  UDP flood attack: It similar to, ICMP flooding attack where an attacker send the UDP packets that are large that the server can handle then the server is so busy that it cannot even analyze normal traffic.

The following table contains the list of rules present in intrusion detection systems and it also contains list of rules for attacks.

- TELNET ATTACK
  There are three types of telnet attack telenet communicating attack, telnet brute force attack and telnet denial of service attack.
  TELNET brute force attack: Telnet protocols can be used to get control of network switches by the attackers. If we set password for vty lines to get access to the switch it is still not secure. The vty lines password only helps the switch from getting unauthorized access. This not an effective method for securing vty lines because there are a lot of tools that can able to launch brute force password crack attack against the vty lines.
  https://howdoesinternetwork.com/2011/telnet-attacks

- UDP ATTACK
  UDP flood attack we send UDP packets to a specified port to the the targeted system. To determine the requested application, the victim system processes the incoming data. In case of absence of the requested application on the requested port, the victim system sends a "Destination unreachable" message to the sender (attacker). In order to hide the identity of the attacker, the attacker often spoofs the source IP address of the attacking packets. UDP flood attacks may also depletes the bandwidth of network around the victim's system. Thereby, the systems around the victim are also impacted due to the UDP flooding attack.

- FTP ATTACK
  In ftp attack in which the unauthorized user uses ftp protocol to get access to the other user's data.

Table 4-1: Different attacks that can be detected in IDS [7]

Rule Types	SNORT	Suricata	OSSEC	Rule Types	Snort	Suricata	OSSEC
bad-traffic	yes	yes	yes	attack-responses	yes	yes	yes
exploit	yes	yes	yes	oracle	yes	yes	yes
scan	yes	yes	yes	mysql	yes	yes	yes
finger	yes	yes	yes	snmp	yes	yes	yes
ftp	yes	yes	yes	smtp	yes	yes	YES
telnet	yes	yes	yes	imap	yes	yes	yes
rpc	yes	yes	yes	pop2	yes	yes	no
rservices	yes	yes	yes	pop3	yes	yes	yes
dos	yes	yes	yes	nntp	yes	yes	yes
ddos	yes	yes	yes	web-attacks	yes	yes	yes
dns	yes	yes	yes	backdoor	yes	yes	yes
tftp	yes	yes	yes	shellcode	yes	yes	yes
web-cgi	yes	yes	yes	policy	yes	yes	yes
web-coldfusion	yes	yes	yes	porn	yes	yes	yes
web-iis	yes	yes	yes	info	yes	yes	yes
web-frontpage	yes	yes	yes	icmp-info	yes	yes	yes
web-misc	yes	yes	yes	virus	yes	yes	yes
web-client	yes	yes	yes	chat	yes	yes	yes
web-php	yes	yes	yes	multimedia	yes	yes	yes
sql	yes	yes	yes	p2p	yes	yes	yes
x11	yes	yes	yes	spyware-put	yes	yes	no
ssh	yes	yes	yes	specific-threats	yes	yes	yes
icmp	yes	yes	yes	experimental	yes	yes	yes
netbios	yes	yes	yes	content-replace	yes	yes	yes
misc	yes	yes	yes	voip	yes	yes	yes

4.2 Tools used in experimentation

- *Network generation tools:*

In this these are the tools that are used to conduct experimentation

- *Hping3*
  This an open source network traffic generator that can generate TCP/IP packets. It relies on ping program with ICMP connection. With this we can what types of traffic we can send to host and different speeds and number of packets. It can be used for DDOS attack.
  Features of Hping3:
  - It is useful for testing firewall rules.
  - It has advanced port scanning features.
  - It can be used to measure network performance using protocols.
  - It can transfer files even between complicated firewall
  - It can remote fringerprint operating system.
  -

hping2 [**-c** *count*] [**-i** *wait*] [**--fast**] [**-I** *interface*] [**-9***signature*] [**-a** *host*] [**-t** *ttl*] [**-N** *ip* id] [**-H** *ip* protocol] [**-g** *fragoff*] [**-m** *mtu*] [**-o** *tos*] [**-C** *icmp* type] [**-K***icmp* code] [**-s** *source* port] [**-p[+][+]** *dest* port] [**-w** *tcp* window] [**-O** *tcp* offset] [**-M** *tcp* sequence number] [**-L** *tcp* ack] [**-d** *data* size] [**-E** *filename*] [**-e** *signature*] [**--icmp-ipver** *version*] [**--icmp-iphlen** *length*] [**--icmp-iplen** *length*] [**--icmp-ipid** *id*] [**--icmp-ipproto** *protocol*] [**--icmp-cksum** *checksum*] [**--icmp-ts**] [**--icmp-addr**] [**--tcpexitcode**] [**--tcp-timestamp**] [**--tr-stop**] [**--tr-keep-ttl**] [**--tr-no-rtt**] hostname

Table 4-2 example command of Hping3

- *Nmap*

It is an open source tool for network auditing, check traffic and security. Many find it useful for checking the network, for monitoring the host, time taken for the service to complete. It is also used for raw packets to determine the host that are available on network and the port hat are open to receive data. It can scan large networks and can also determine small hosts. NMAPcan also be used as an attack generator. It is a graphical user interface.

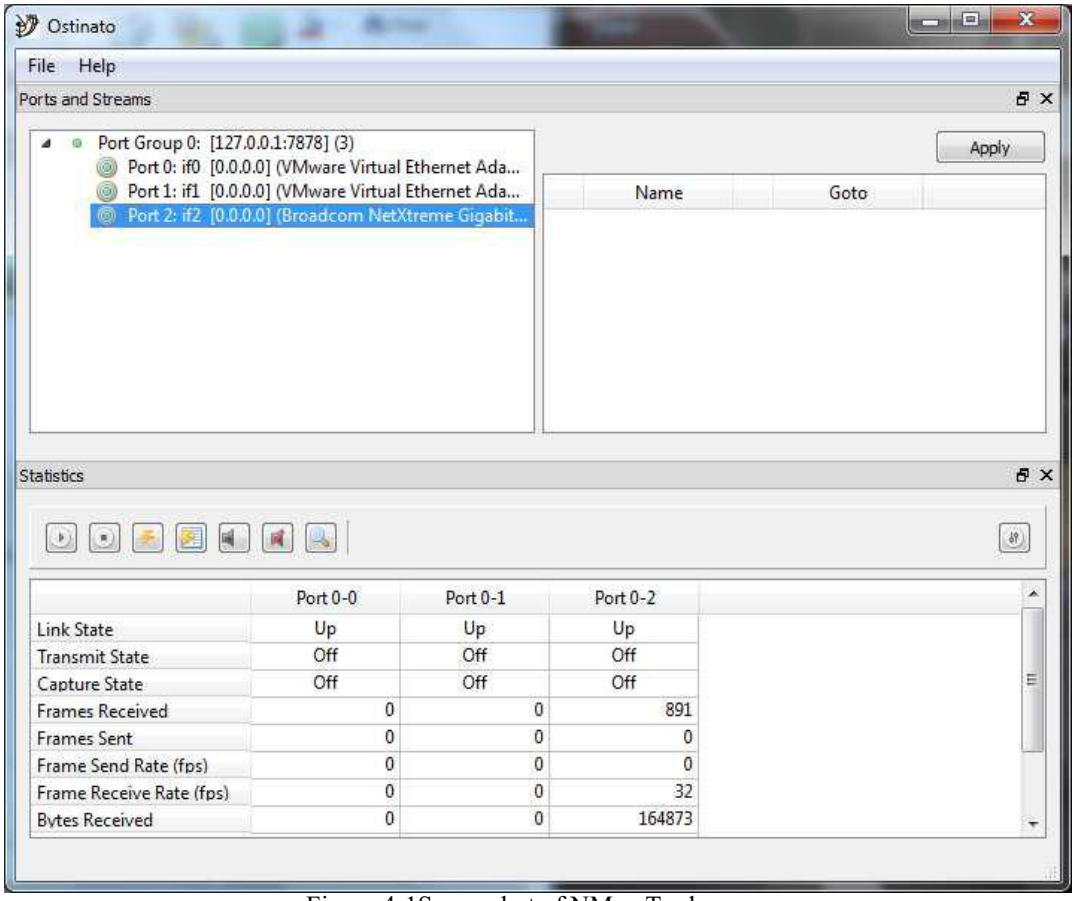

Figure 4-1Screenshot of NMap Tool

- *Portsentry*

  It is used to scan the port and incoming packets and establishing connections. It is used to create logs of the analyzed packets. We use portsentry with OSSEC.

4.3 Experimentation setup

The figure discusses the testing environment that we are using conduct our simulation on. The main parts of this environment are network traffic generator it is used to generate the normal traffic and malicious traffic by using Hping3. network switch, Target server is server on which we deploy attack on. Intrusion detection system we are using three intrusion detection system OSSEC, Suricata and Snort. The packets are generated by hping3 and then sends them to the intrusion detection system for analysis.

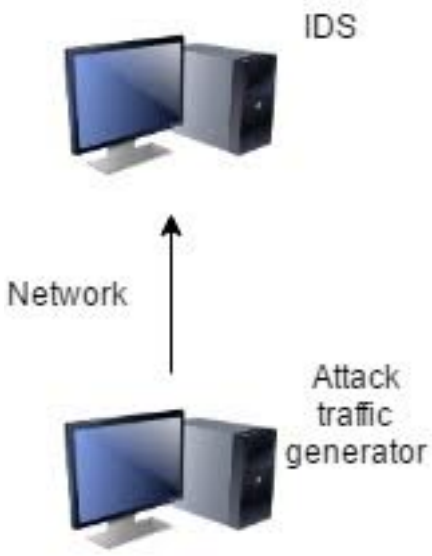

Figure 4-2 Virtual environment to test IDS

4.4 Hardware and Software

The below tables discuss the hardware and software requirements.

Hardware	Software
Macbook Md101	Hping3
Virtual box	Putty, UBUNTU
Intel i5 processor	NMAP, WINE app
RAM 2GB, HDD 60GB	OS x El capitan

Table 4-3 Hardware and software requirements of attack server

Hardware	Software
Macbook Md101	Snort, Bro, OSSEC
Virtual box	Putty, UBUNTU
Intel i5 processor	NMAP, portsentry, WIneapp
RAM 2GB, HDD 30GB	OS x El capitan

Table 4-4 Hardware and software requirements of IDS server

4.5 Performance Metrics

We choose this three IDS because they are most popularly used and they are all signature based detection systems that can configurable with a rule set. They have some common metrics that can be used to measure their performance. They are all ubuntu compatible. And also respond DOS.

- CPU usage: measured in percentage.
- RAM usage: measured in percentage.
- Through put: percentage or number of packets.
  Alerts generated: They are measured in number generated.

5 RESULTS AND ANALYSIS

This chapter focuses on the experiment implementation of three intrusion detection systems. This experiment helps us determine accuracy and performance of the intrusion detection system. The experiment system contains malicious and non- malicious traffic which are generated by software component such as hping3. The traffic is attacks the target server as shown in the figure. These traffic are monitored by intrusion detection systems such as Suricata, snort and OSSEC. The traffic is sent from the generator to the receiver where the performance of intrusion detection can be measured. The packet generator can generate malicious attacks which are sent through the IDS to measure the accuracy of IDS.

Performance metrics that are used in determining the performance of intrusion detection system are as below.

- Packet loss
  The packet loss id defined as number packets that are dropped by intrusion detection system is equal to number packets sent by network generator minus total packets received by intrusion detection system.
- Number of alerts generated:
  It is determined by number alerts that are generated by malicious traffic. These are counted using log files in intrusion detection system.
- CPU and memory usage of intrusion detection system:
  This can be monitored using commands like top to monitor the CPU and memory usage.

5.1 Experiment 1 and result

This experiment evaluates the intrusion detection system under different speed of packets under normal traffic. The below tables represents the memory usage, CPU usage and packet loss. For intrusion detection system snort, Suricata and OSSEC.

a) Memory usage:

Table 5-1: Tabular Result of Memory usage

IDS	PERCENTAGE (%)
Snort	34.8
Suricata	24.7
OSSEC	31

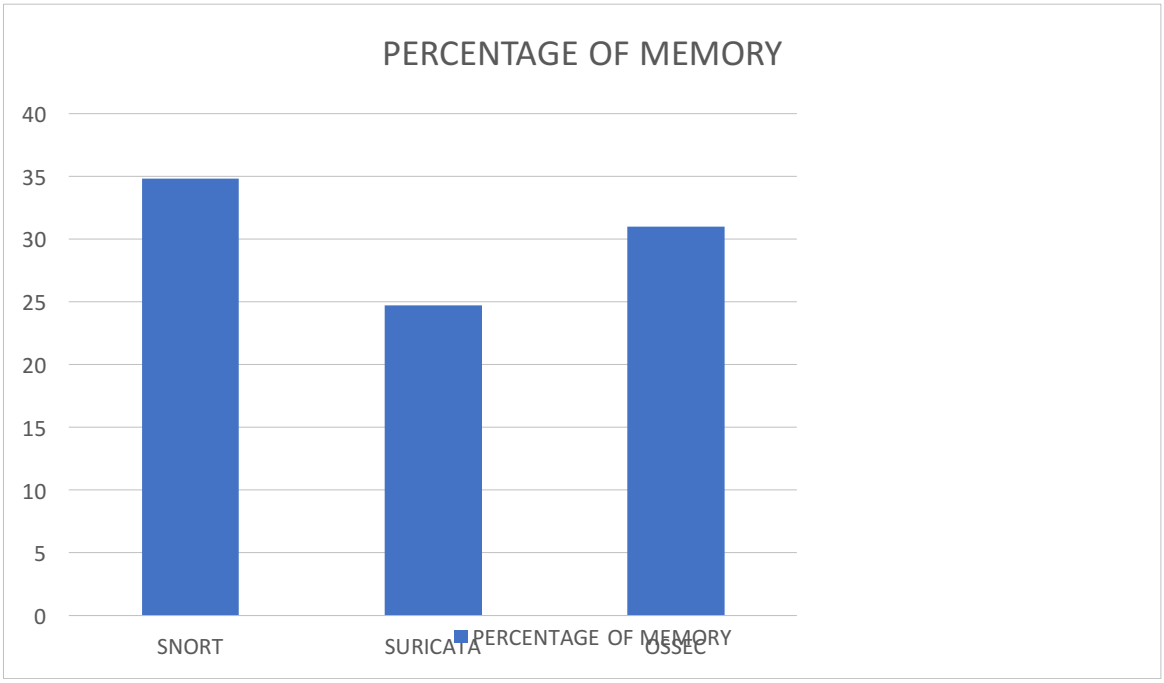

Figure 5-1: Bar Chat representation of Memory usage

b) CPU usage – normal traffic:

Table 5-2: CPU usage – normal traffic

Traffic rate	percentage	percentage	percentage
Packets per sec	SURICATA	SNORT	OSSEC
100	5.6	4.4	3.5
500	8.2	6.3	7.2
1000	12.3	14.6	16.2
2000	18.3	26.4	21.3
5000	31.5	34.5	30.4
10000	47.3	43.6	38.2
20000	56.0	54.2	51.2
50000	65.4	60.8	56.9

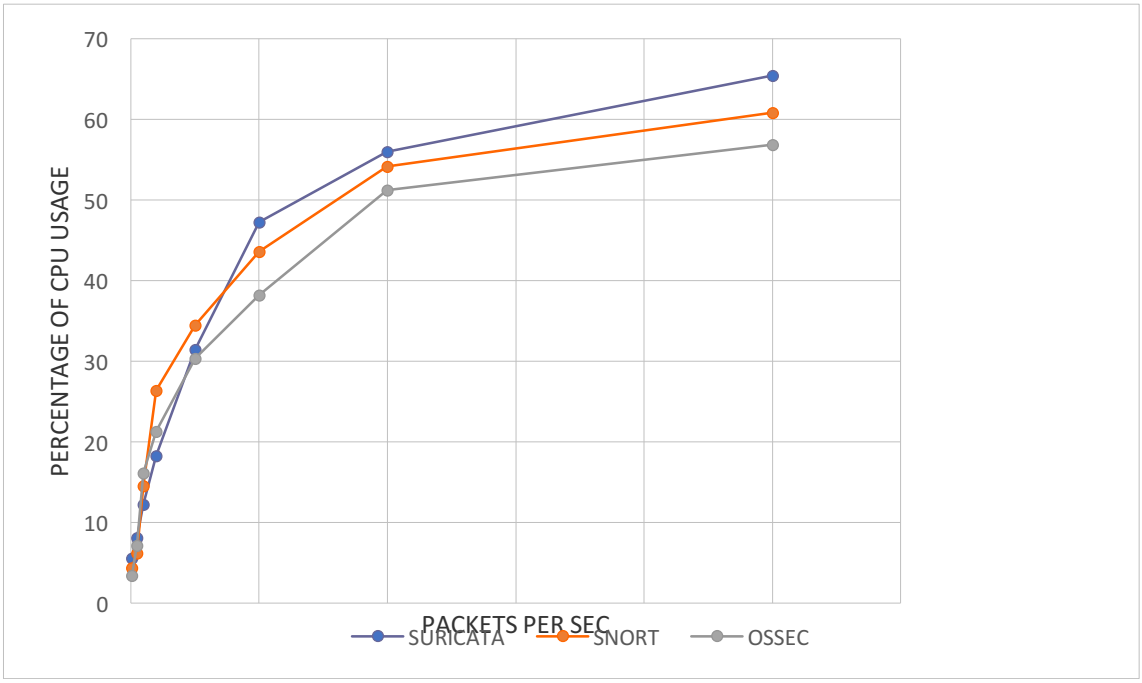

Figure 5-2: CPU usage of snort, suricata and OSSEC

c) Packet Loss

Table 5-3: Packet Loss rate in Suricata Snort and OSSEC

Traffic rate	percentage	percentage	percentage
Packets per sec	SURICATA	SNORT	OSSEC
100	0	0	0
500	0	0	0
1000	0	0	0
2000	0	0	0
5000	2.4	3.6	2.8
10000	9	12.3	11.2
20000	20.3	23.5	18.6
50000	41	39.8	36.9

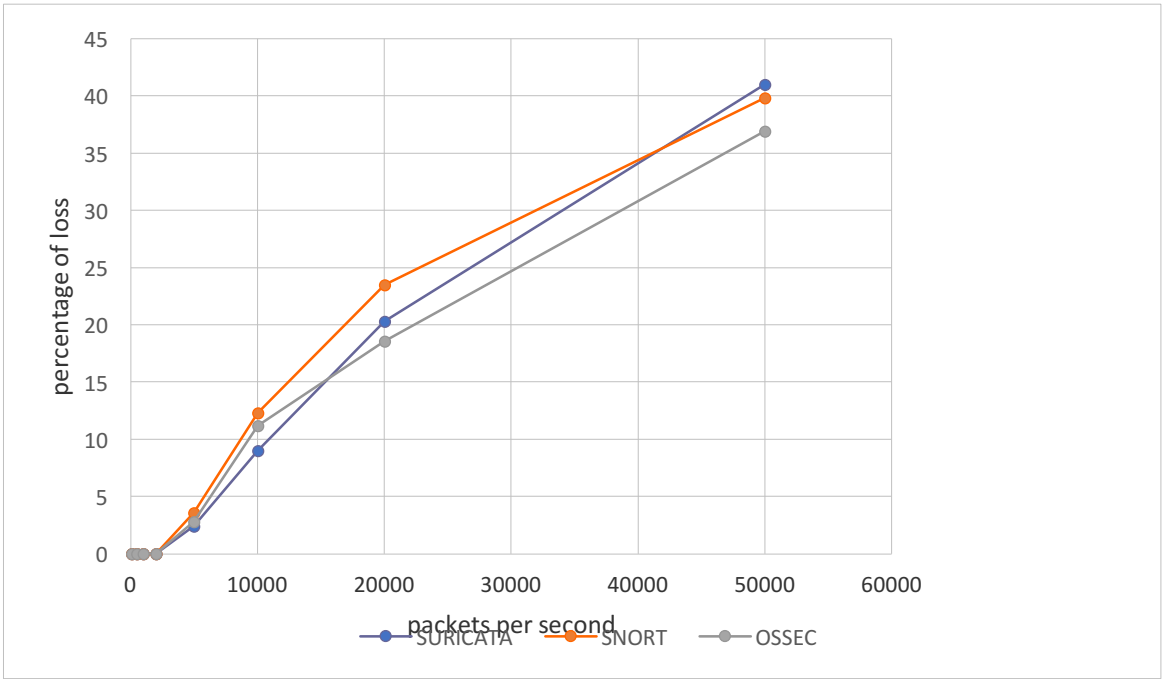

Figure 5-3: Comparison packet loss among snort, suricata and OSSEC

5.2 Experiment 2 and results

This experiment evaluates the intrusion detection system under malicious traffic in different speeds. The below tables represent the memory usage, number f alerts, CPU utilization and packet loss. Malicious Traffic is generated using the following commands:

hping3 -S -P -U -i uxxxx -V --rand-source 192.168.1.102

-S set the SYN TCP Flag

-U set the URG TCP Flag

-P All Ports

-V verbose

--rand-source will send packets with random source address.

-i uX - representing the amount of time should wait before sending the new packet.

-i u10000 will send 10 packets per second.
-i u1000 will send 100 packets per second.
-i u200 will send 500 packets per second.
-i u100 will send 1000 packets per second.
-i u50 will send 2000 packets per second
-i u20 will send 5000 packets per send.
-iu10 will send 10000 packets per second.

5.2.1 TCP attack

a) Memory Usage

Table 5-4: Memory Usage under TCP attack by IDS

IDS	PERCENTAGE(%)
Snort	38.8
Suricata	29.7
OSSEC	37

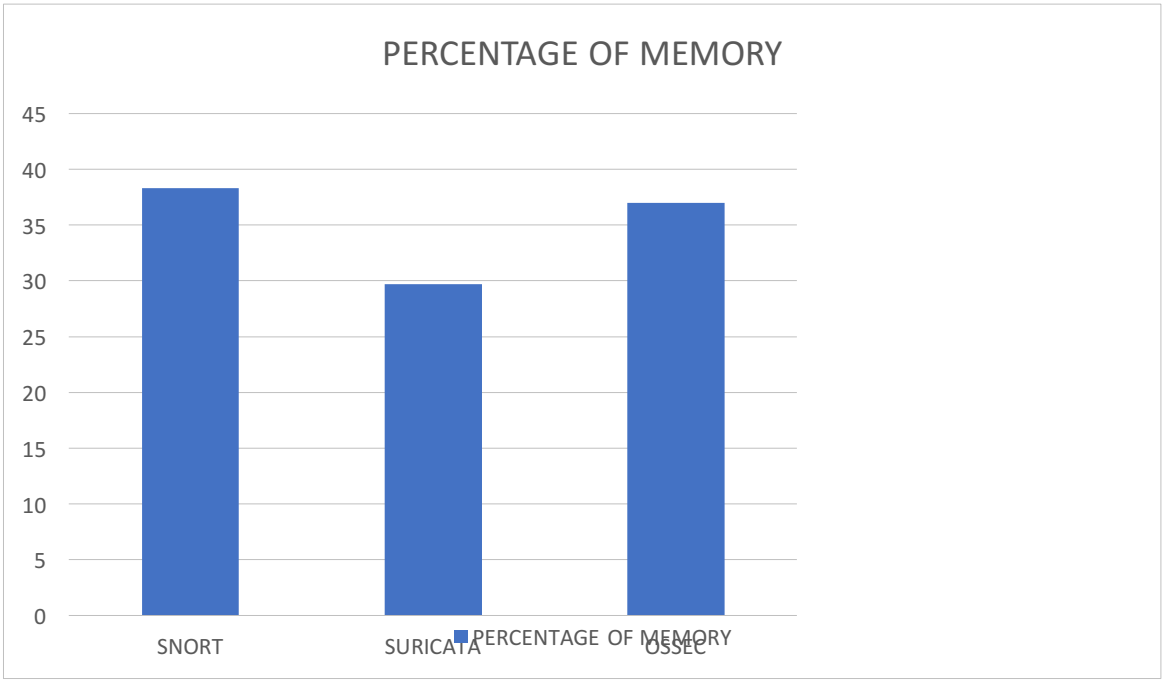

Figure 5-4: Memory Usage under TCP attack by IDS

b) CPU usage – malicious traffic

Table 5-5: Percentage of cpu usage under TCP attack by suricata, snort and OSSEC

Traffic rate	Percentage (%)		
Packets per sec	SURICATA	SNORT	OSSEC
100	7.4	6.4	5.6
500	11.2	14.3	13.5
1000	18.3	19.6	21.2
2000	28.5	31.4	34.3
5000	38.7	39.1	36.4
10000	51.3	47.3	45.2
20000	59.6	60.2	59.2
50000	70.4	72.2	65.9

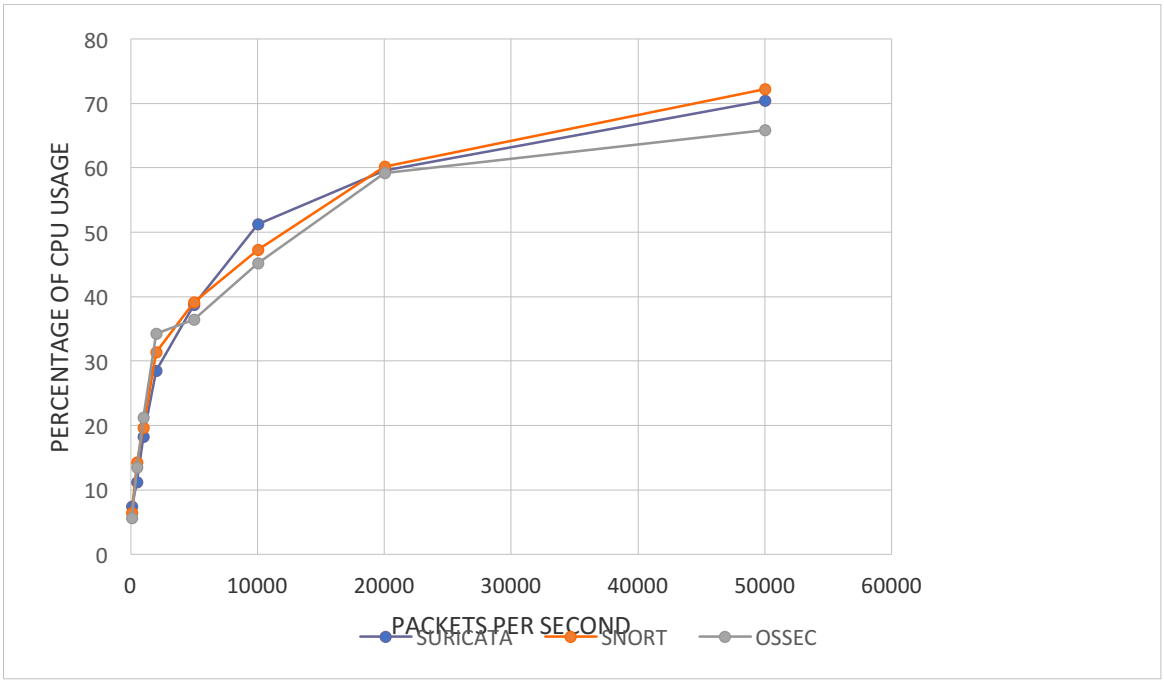

Figure 5-5: Percentage of cpu usage under TCP attack by suricata, snort and OSSEC

c) Packet loss:

Table 5-6: Percentage of packets lost by Suricata, snort and OSSEC under TCP attack

Traffic rate	percentage	percentage	percentage
Packets per sec	SURICATA	SNORT	OSSEC
100	0	0	0
500	0	0	0
1000	0.1	0.3	0.3
2000	1.2	1.6	1
5000	4.4	5.6	5.8
10000	10.8	12.8	14.6
20000	23.3	28.5	20.5
50000	45	36.4	40.7

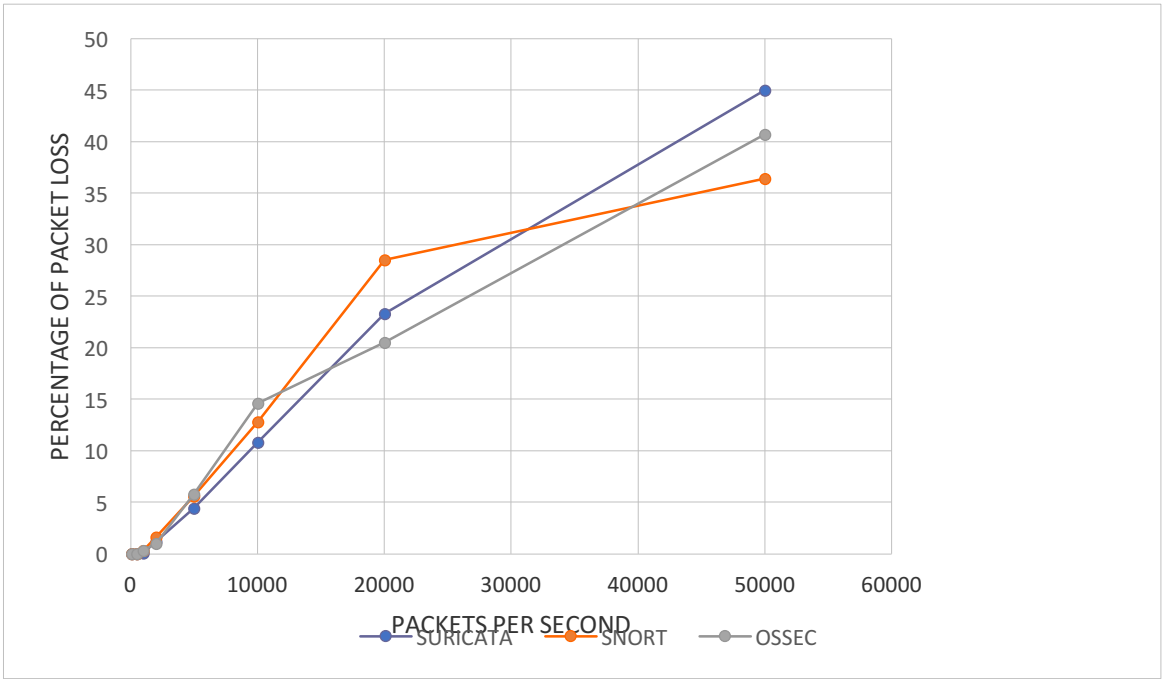

Figure 5-6: Percentage of packets lost by Suricata, snort and OSSEC under TCP attack.

d) Number of alerts

Table 5-7: Number of alerts for suricata, snort and OSSEC

Traffic rate	percentage	percentage	percentage
Packets per sec	SURICATA	SNORT	OSSEC
100	1840	1920	1720
500	1840	1920	1720
1000	1821	1901	1701
2000	1801	1891	1684
5000	1759	1888	1620
10000	1641	1744	1468
20000	1411	1430	1367
50000	1012	1221	1019
Average	1640	1739	1537

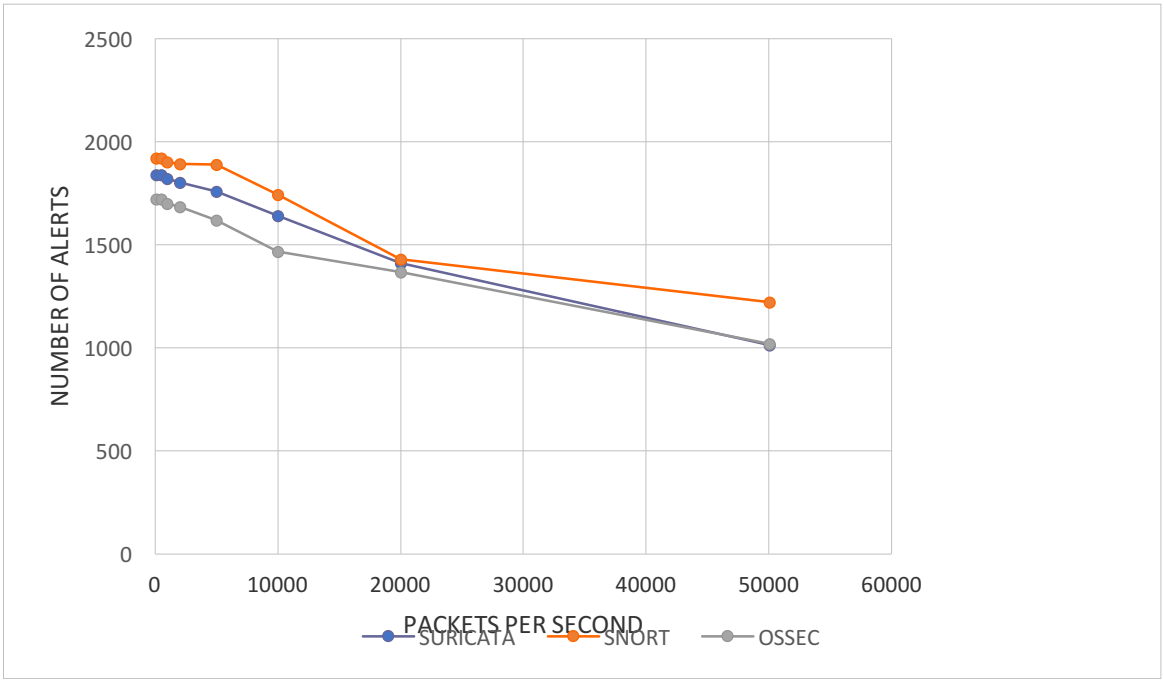

Figure 5-7: Number of alerts for suricata, snort and OSSEC

5.2.2 FTP attack

a) Memory usage

Table 5-8: Percentage of Memory usage by Suricata, snort and OSSEC under FTP attack.

IDS	PERCENTAGE(%)
Snort	39.4
Suricata	28.3
OSSEC	38

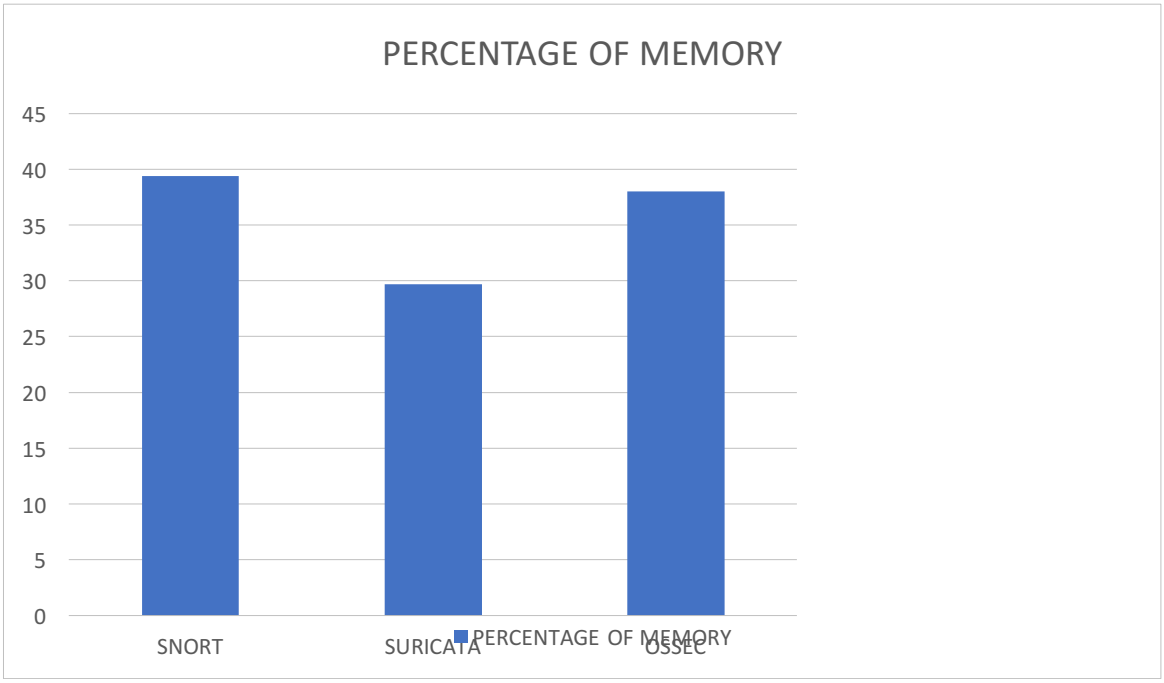

Figure 5-8: Percentage of Memory usage by Suricata, snort and OSSEC under FTP attack.

b) CPU usage – malicious traffic

Table 5-9: Percentage of CPU usage by suricata, snort and OSSEC

Traffic rate	Percentage (%)		
Packets per sec	SURICATA	SNORT	OSSEC
100	8	11	8
500	16	21	15
1000	18	28	21
2000	31	31	30
5000	42	46	36
10000	59	61	62
20000	60	70	64
50000	74	83	70

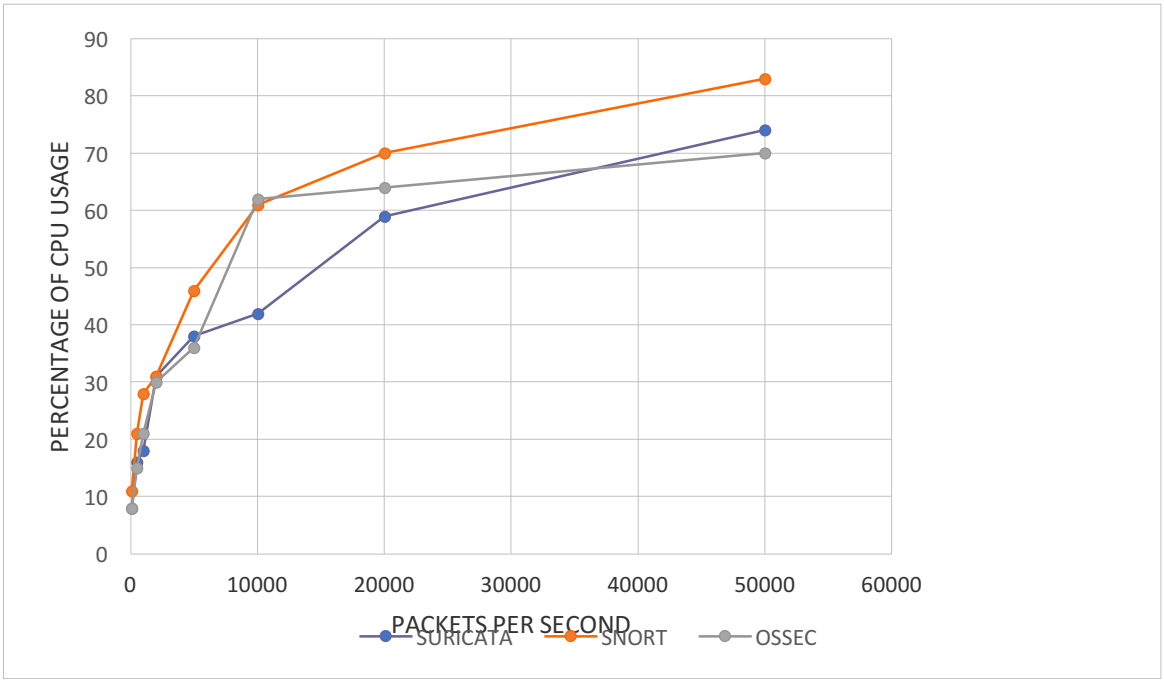

Figure 5-9 Percentage of cpu usage by suricata, snort and OSSEC

c) Packet loss

Table 5-10: Percentage of packets lost by suricata,snort and OSSEC

Traffic rate	percentage	percentage	percentage
Packets per sec	SURICATA	SNORT	OSSEC
100	0	0	0
500	0	0	0
1000	0.4	0.1	0.2
2000	2	0.4	1
5000	4	3	6.2
10000	9.7	12.8	13.2
20000	23	23.3	20
50000	42	30	37

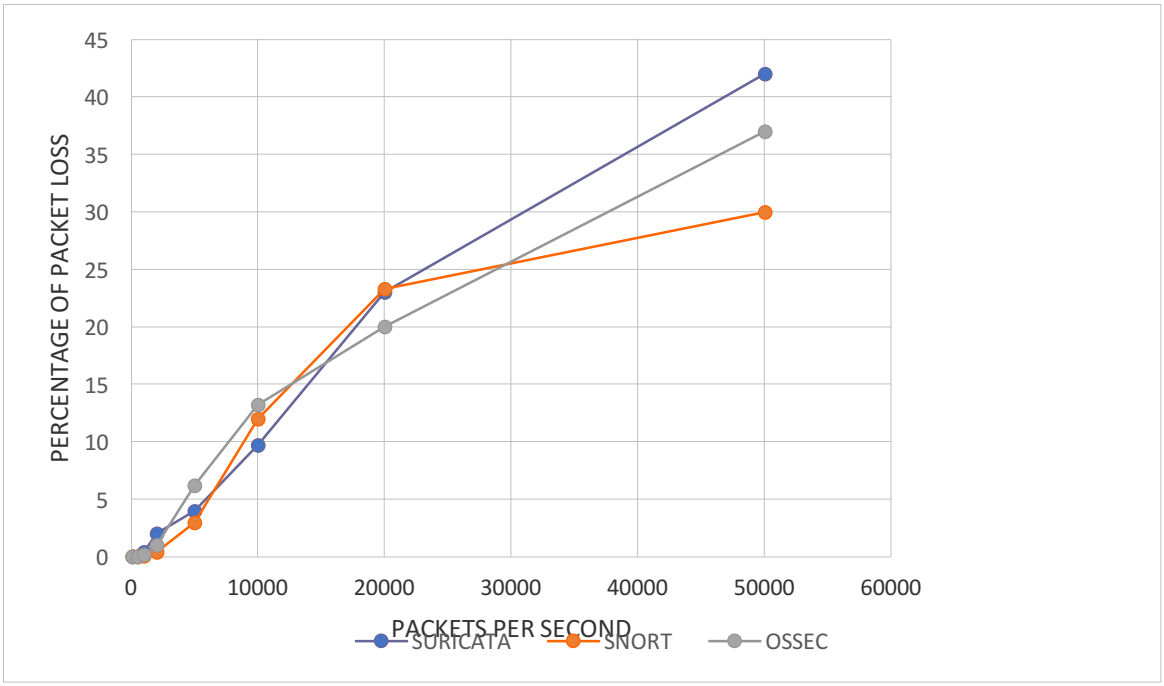

Figure 5-10: Percentage of packets lost by suricata,snort and OSSEC

d) Number of alerts

Table 5-11: Number of alerts for suricata, snort and OSSEC

Traffic rate	percentage	percentage	percentage
Packets per sec	SURICATA	SNORT	OSSEC
100	964	942	1003
500	964	942	994
1000	940	920	982
2000	937	914	947
5000	920	897	917
10000	911	892	905
20000	907	884	902
50000	907	879	895
Average	931	911	943

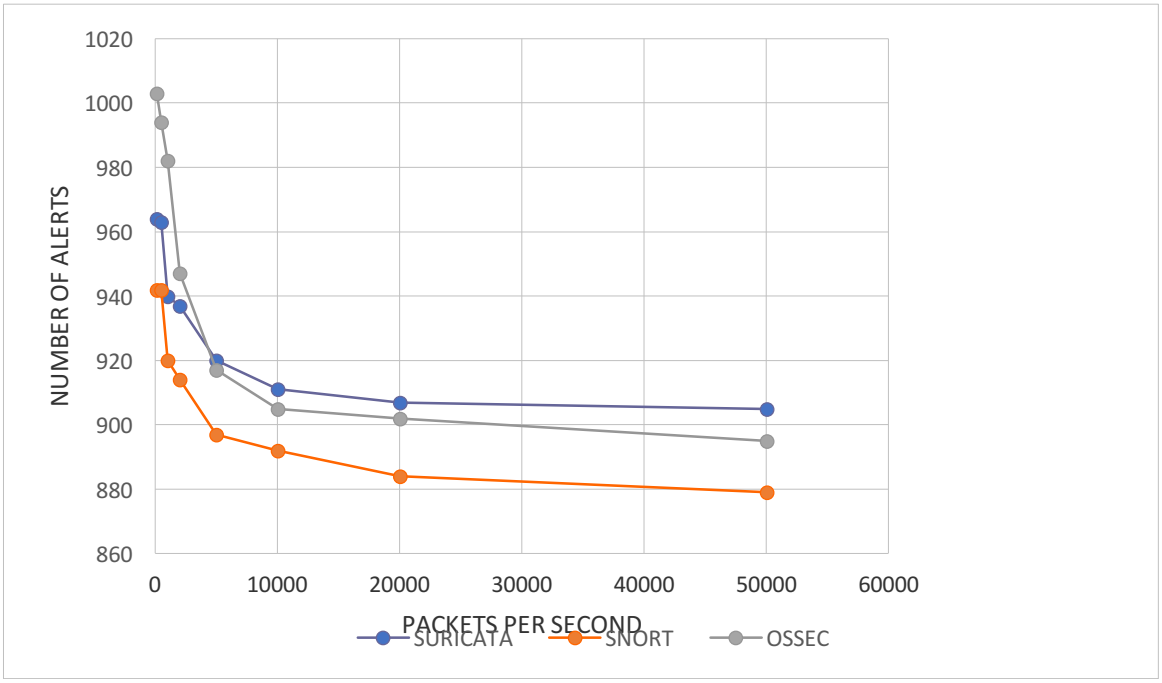

Figure 5-11: Number of alerts for suricata, snort and OSSEC

5.2.3 ICMP Attack

a) Memory usage

Table 5-12: Percentage of Memory usage by suricata, snort and OSSEC

IDS	PERCENTAGE(%)
Snort	18
Suricata	22
OSSEC	19.5

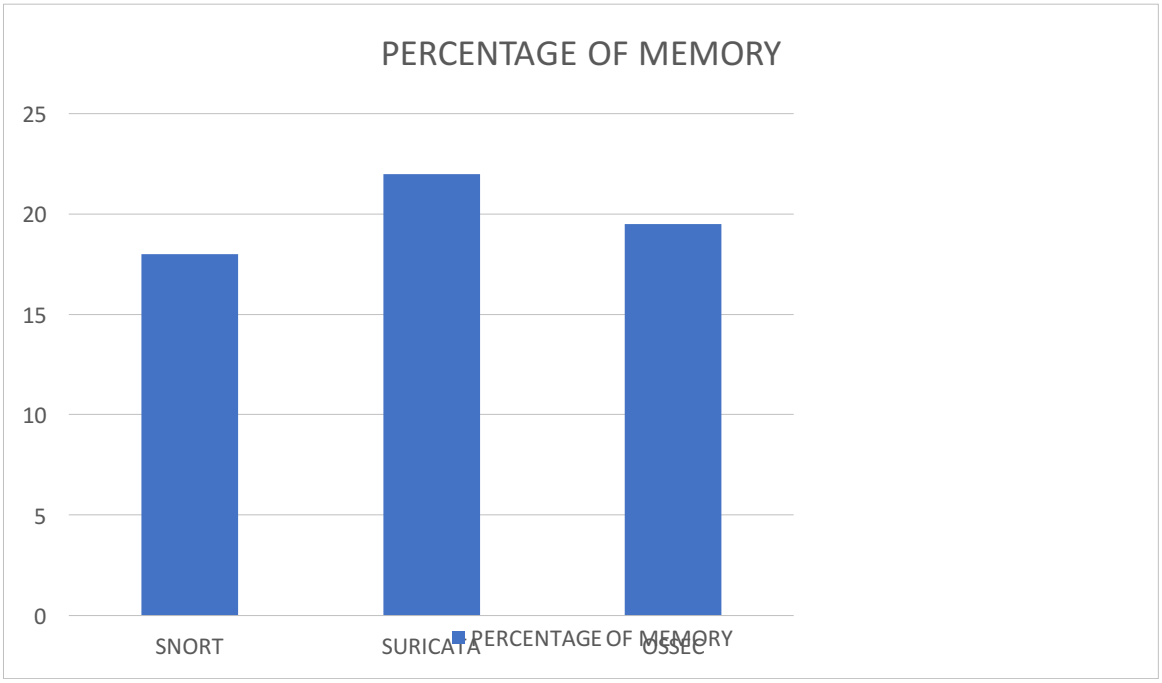

Figure 5-12: Percentage of Memory usage by suricata, snort and OSSEC

b) CPU usage – malicious traffic

Table 5-13: Percentage of CPU usage by suricata, snort and OSSEC

Traffic rate	Percentage (%)		
Packets per sec	SURICATA	SNORT	OSSEC
100	8	13	11
500	10	15	18
1000	19	19.7	21
2000	32	39.2	34
5000	42	41	39.5
10000	47	49.4	44.2
20000	54.5	58	48
50000	59.4	66	55

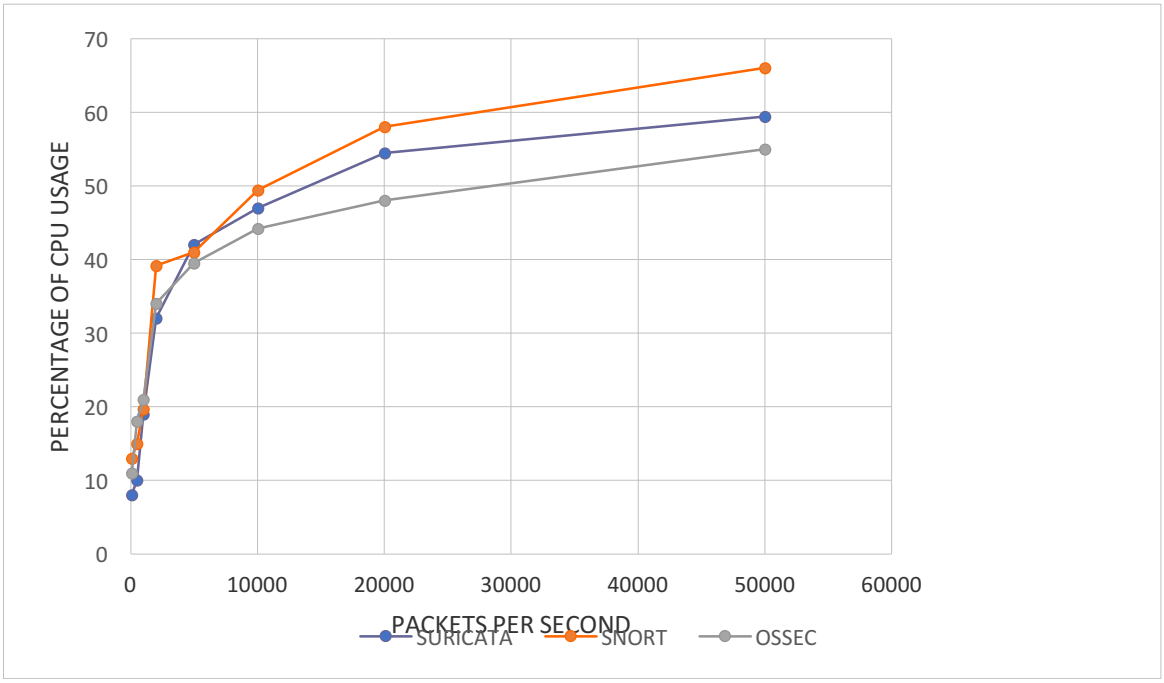

Figure 5-13 Percentage of CPU usage by suricata, snort and OSSEC

c) Packet loss

Table 5-14: Percentage of packets lost by suricata,snort and OSSEC

Traffic rate	percentage	percentage	percentage
Packets per sec	SURICATA	SNORT	OSSEC
100	0	0	0
500	0	0	0
1000	0.1	1	8
2000	8.3	15	11.6
5000	9.5	16.5	18
10000	19	30	25
20000	41.5	34.5	25.5
50000	58	50	35

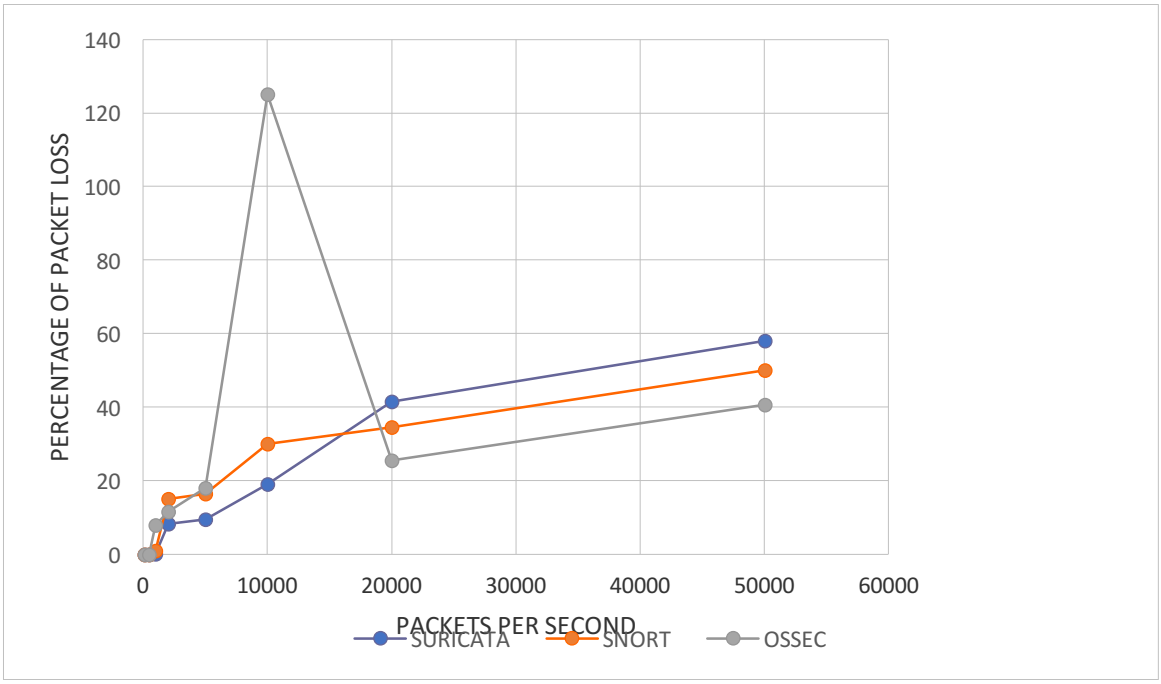

Figure 5-14: Percentage of packets lost by suricata,snort and OSSEC

d) Number of alerts

Table 5-15: Number of alerts for suricata, snort and OSSEC

Traffic rate	percentage	percentage	percentage
Packets per sec	SURICATA	SNORT	OSSEC
100	1963	1824	1947
500	1960	1824	1944
1000	1859	1740	1843
2000	1853	1692	1836
5000	1843	1670	1844
10000	1834	1613	1837
20000	1822	1587	1804
50000	1809	1505	1704
Average	1868	1682	1849

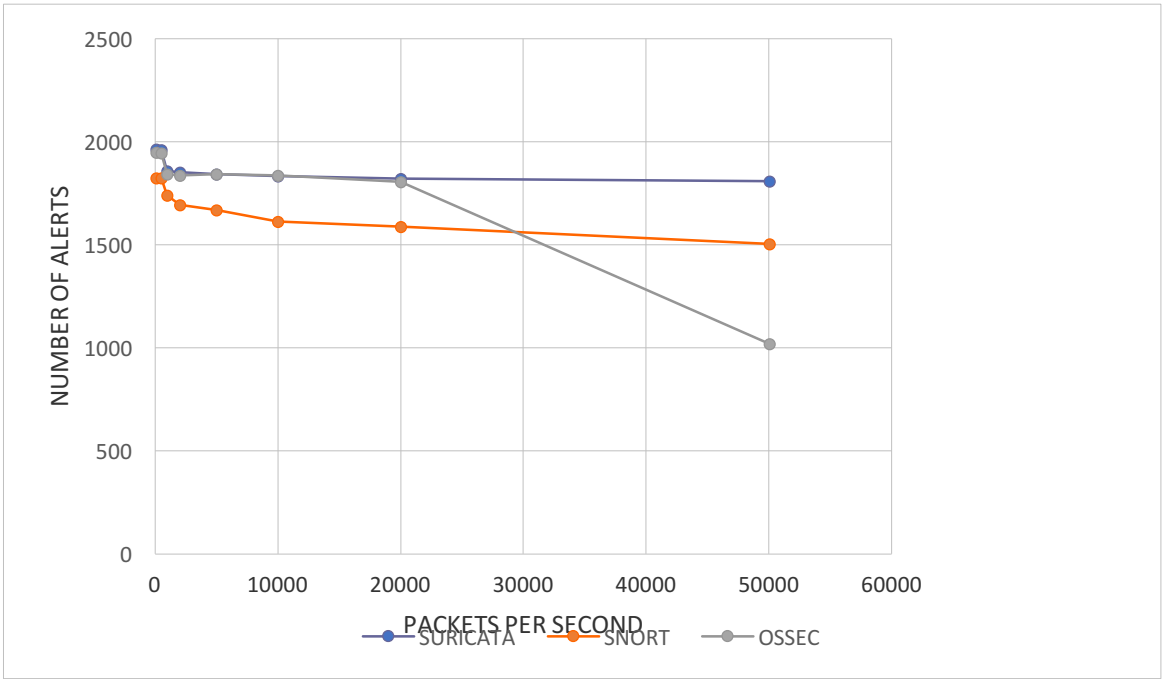

Figure 5-15 Number of alerts for suricata, snort and OSSEC

5.2.4 SCAN Attack

a) Memory usage

Table 5-16: Memory Usage for suricata, snort and OSSEC

IDS	PERCENTAGE(%)
Snort	18
Suricata	21
OSSEC	19.3

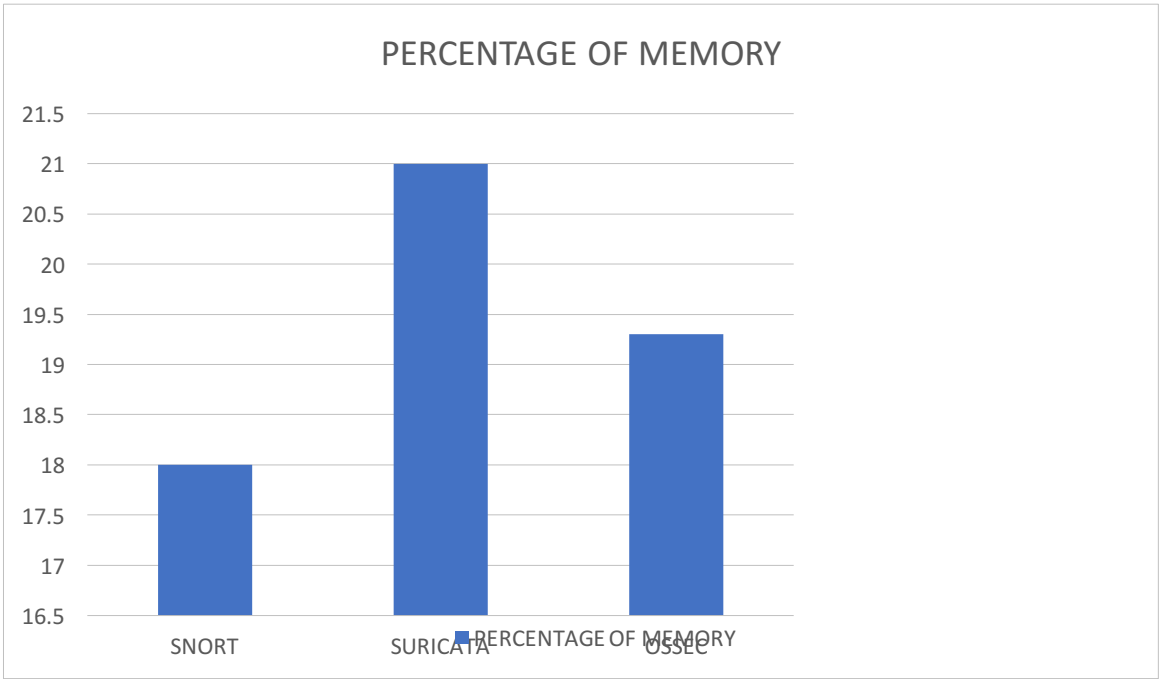

Figure 5-16: Memory Usage for suricata, snort and OSSEC

b) CPU usage – malicious traffic

Table 5-17: Percentage of CPU usage by suricata, snort and OSSEC

Traffic rate	Percentage (%)		
Packets per sec	SURICATA	SNORT	OSSEC
100	7	10	9
500	10	15	16
1000	15	27	23
2000	33	39	32
5000	41	46	40.5
10000	45	54	42
20000	52	57	48
50000	64	62	52

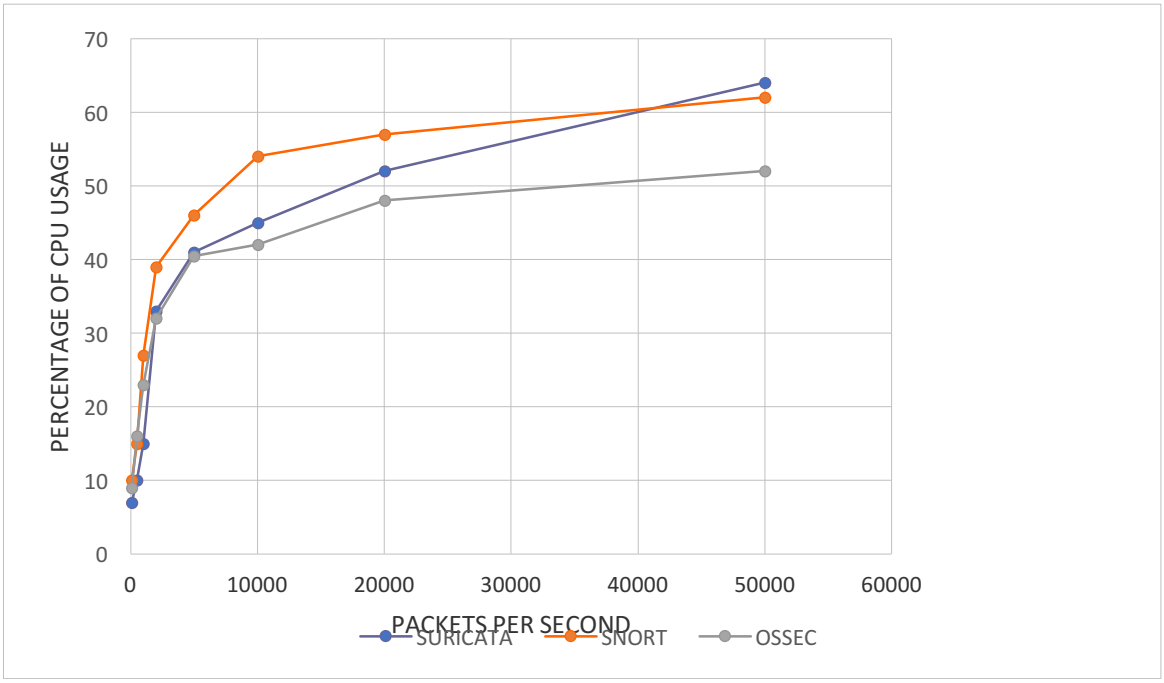

Figure 5-17 Percentage of cpu usage by suricata, snort and OSSEC

c) Packet loss

Table 5-18: Percentage of packets lost by suricata,snort and OSSEC

Traffic rate	percentage	percentage	percentage
Packets per sec	SURICATA	SNORT	OSSEC
100	0	0	0
500	2.3	10.5	11
1000	4.6	11.5	13.8
2000	14.5	19.3	23.4
5000	15.4	22.4	25.4
10000	22.7	26.5	26
20000	37	28	29
50000	48	38.6	42.5

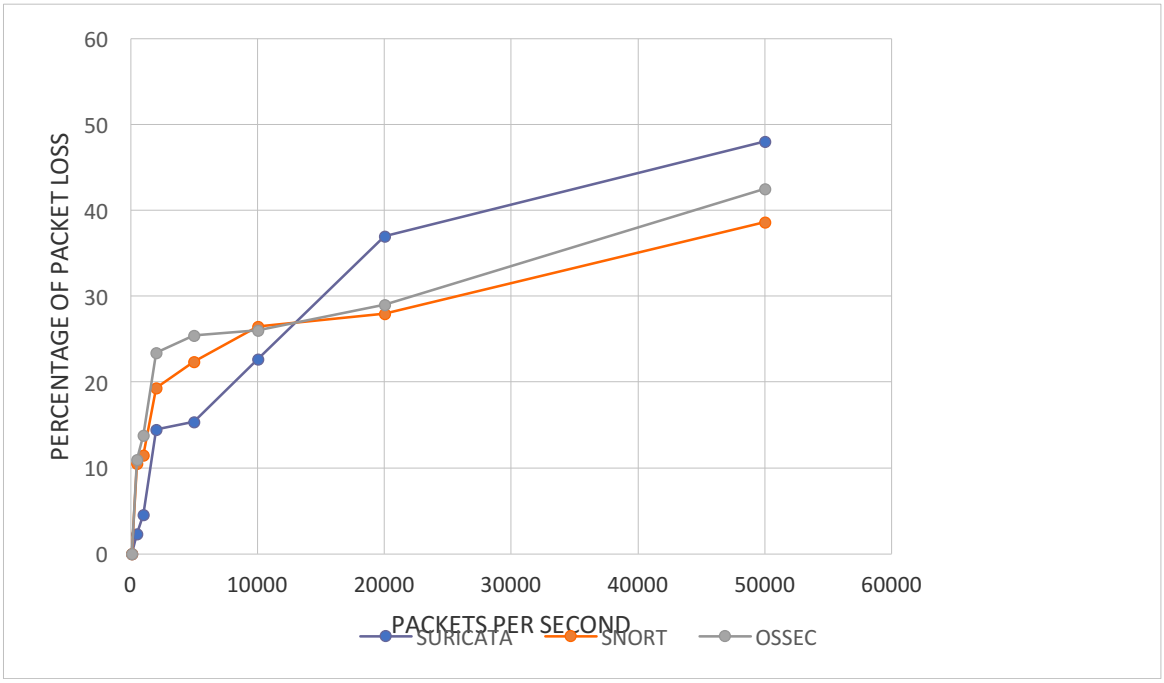

Figure 5-18 Percentage of packets lost by suricata,snort and OSSEC

d) Number of alerts

Table 5-19: Number of alerts for suricata, snort and OSSEC

Traffic rate	percentage	percentage	percentage
Packets per sec	SURICATA	SNORT	OSSEC
100	1379	1124	1541
500	1379	1124	1541
1000	1357	1045	1512
2000	1289	1025	1425
5000	1253	1003	1379
10000	1194	935	1865
20000	1104	904	1857
50000	954	873	1784
Average	1229	1004	1342

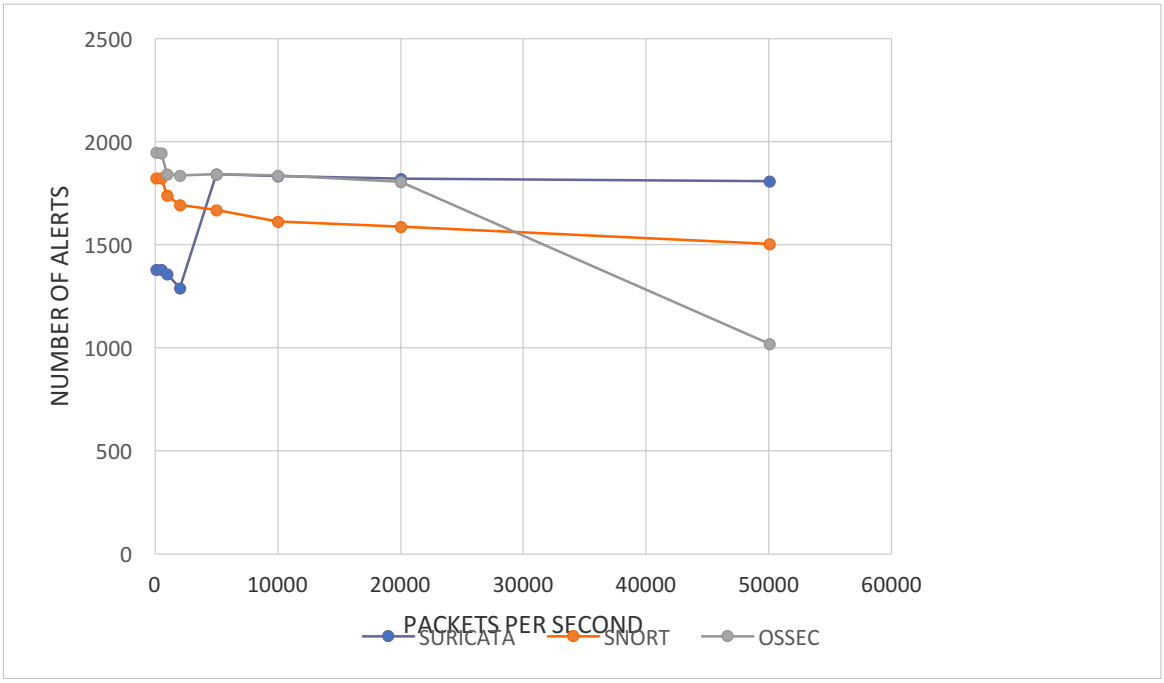

Figure 5-19 Number of alerts for suricata, snort and OSSEC

5.3 Experiment 3

The main objective of the experiment is to know which has better performance on the given malicious traffic detection. This can be told by how accurate a IDS can detect intrusions in the form of alerts. This can be determined by subtracting no alerts generated by all rules to only one rule particular attack rule. The percentage of missed alerts is calculated by following formula.

Let
A = number alerts generated by all rules
B = number of alerts generated by particular attack

Rule:

Missed alerts percentage = (A-B) / B * 100
Accuracy = 100 – percentage of missed alerts

The below table shows the number of alerts and accuracy of particular alerts.

Table 5-20: Number of alerts and accuracy of suricata

Attack type	Suricata		
	1 Rule set	All rules	Accuracy rate %
DOS	1228	1640	66.44
FTP	840	931	89.16
ICMP	1438	1868	70.09
SCAN	1032	1229	80.91

Table 5-21: Number of alerts and accuracy Snort

Attack type	Snort		
	1 Rule set	All rules	Accuracy rate %
DOS	1254	1739	61.32
FTP	845	911	92.19
ICMP	1259	1682	61.47
SCAN	858	1004	82.72

Table 5-22: Number of alerts and accuracy of bro

Attack type	OSSEC		
	1 Rule set	All rules	Accuracy rate
DOS	1268	1537	78.78
FTP	878	943	92.5
ICMP	1427	1849	70.4
SCAN	1132	1342	81.44

The increase in packet rates affects the performance of intrusion detection system. The Suricata has low packet loss in tcp flood. Whereas bro and snort almost have same packet loss in tcp traffic.

When three intrusion detection systems have packet loss at high rates the CPU utilization also increases. That means packet rated effects the intrusion detection systems performance. Suricata when compared has the highest CPU utilization. Compare to other IDS in all traffic OSSEC has high CPU and packet loss. But OSSEC has best accuracy and can better detect attacks compared other two IDS.

6 DISCUSSION

This chapter describes about the answer to thesis question and its limitations in section 6.1 and 6.2

6.1 Answers to Research Questions

RQ1: What are the state of the art solutions for Intrusion Detection System in cloud computing?

Literature review has been done to know the state of art solution for existing intrusion detection system. The section 1.1, 1 and section 2.1 discusses the state of intrusion detection system. The propose of this question is to review the intrusion detection system available that can be used for cloud computing. The cloud computing architecture, performance. We discuss about the snort, Suricata and Ossec with their architecture and their rules. There are host based intrusion detection systems, network based IDS, Hypervisor based IDS, Distributed IDS. The method that are used for detection of intrusion are Signature based IDS, Anomaly based detection and Hybrid detection technique.

RQ2: What are the main types of cloud attacks that can be detected using Intrusion Detection System?

Literature review has been done to know different attacks that can be detected using intrusion detection system. The intrusion detection system uses different set of rules to detect the malicious attacks. We have also compared the different attacks that can be detected by the selected intrusion detection systems. The section 4.1 contains all the attacks that can be detected using intrusion detection system. Some of the attacks that are described are BAD traffic attack, scanning attack, Denial of service attack, penetration attack, ICMP attack, TCP attack, ARP flooding attack, UDP. These all attacks are most defined attack in papers that are reviewed.

RQ3: What is the performance of different Intrusion Detection System?

Experiments are conducted ten times and these are the average results from the experiment. Each experiment is done for both normal traffic and malicious traffic. The result that are considered for measuring the performance of IDS are CPU usage, memory usage, packet loss and number of alerts generated. The IDS that are used are snort, suricata and OSSEC. The results are described in the sections 5.1 and 5.2. The traffic for both malicious and normal traffic are generated through hping3 software and the experiment is conducted in an virtual environment. Snort showed better performance in alerts identification and OSSEC in performance of IDS. This results indicated that alerts are low when the traffic rates high are which indicates this is due to the packet loss. Overall OSSEC provides better performance. And Snort provides better performance and accuracy for alert detection.

6.2 Threats of validity

6.2.1 Internal Validity

The internal validity is change in dependent variable which are based by independent variable

- In this experiment we used limited number of packets so it can replicate the real time traffic generation.
- For capturing packets we use wireshark It usually capture the packets that are sent. Sometimes it drops some packets which may affect the result.

6.2.2 External validity

By [21] "*external validity are conditions that limit our ability to generalize the result of experiment to industrial practice*". Thus the external validities the experiment are as follows.

- The experiment is done in a virtual environment. Thus the results achieved may not be similar to actual cloud environment. Thus this thesis is limited to generalized virtual environment.
- In the experiment Hping3 generates the attack traffic. Thus the characteristics of the packets that generated are related to the tools. Thus results of experiments are confined the traffic Hping3 generated.
- Thus experiment is conducted on on available environments and data given. So this results cannot generalized to other cloud environments.

7 CONCLUSION AND FUTURE WORK

This chapter discusses the thesis contribution and the limitations and threats of the work that we have done and also future work.

7.1 Contribution of Thesis

- Per the survey of cloud security alliance [16] DoS attacks are one of the top most security threats for Cloud Computing. IDS technologies are most effective way to detect those attacks in the cloud environment [16] and alert both the network administrators of cloud and preventive security control in cloud architecture. There are many open source IDS tools which can be deployed based on their detection methodologies and technologies. Among them, the top most used IDS tools are Snort, Suricata and OSSEC [16].

- The comparative analysis of the tools will help the cloud service providers to choose the best tool for their data centres. But there is a lack of empirical evaluation between Snort, Suricata and OSSEC tools[7].

- This thesis helps the cloud service providers to analyse the best tool that can be installed in their environment to protect their data centres against DoS attacks.

7.2 Conclusion

This thesis is analysis of three IDS tools namely Snort, OSSEC and suricata. They are different in terms of IDS and have similar characters like architecture, network traffic, sensors, rules, detection unit and packet analyzer. Our objective of this thesis is to analyze the performance of the three proposed IDS in DOS attack. We set up the environment accordingly.

We measured the IDS in normal and malicious traffic. And we used SYN attack to analyze the IDS in malicious traffic and set of rules that are used are taken from official websites. WE measured the performance with CPU utilization and packet loss. The alerts determine how accurate the IDS is detecting the attack.

From the experiment, we outline following results.

- Suricata and snort gave similar results in the performance, but Suricata performs a little better compared to Snort.
- OSSEC performs the best with low CPU usage and packet loss.
- When it comes to alerts generation the OSSEC generates more alerts than SNORT and Suricata.
- The second best in detecting the attacks is Suricata and then Snort.
- We used the same traffic to evaluate all three traffic respectively.

7.3 Future work

The evaluation of performance of IDS under attacks is a difficult task because their different IDS that are developed and rules that are being updated. The number of attacks are also increasing. Thus, experiment should be conducted to evaluate the performance of IDS. So, we can choose the effective IDS tool to battle against the malicious attacks.

REFERENCES

[1] B. Brumen and J. Legvart, "Performance analysis of two open source intrusion detection systems," *2016 39th Int. Conv. Inf. Commun. Technol. Electron. Microelectron.*, pp. 1387–1392, 2016.

[2] N. B. Moe, T. Dingsøyr, and T. Dybå, "Understanding self-organizing teams in agile software development," *Proc. Aust. Softw. Eng. Conf. ASWEC*, no. APRIL, pp. 76–85, 2008.

[3] N. A. Premathilaka, A. C. Aponso, and N. Krishnarajah, "Review on state of art intrusion detection systems designed for the cloud computing paradigm," *2013 47th Int. Carnahan Conf. Secur. Technol.*, pp. 1–6, 2013.

[4] M. Sharma, A. Kaushik, A. Sangwan, and M. Scholor, "Performance Analysis of Real Time Intrusion Detection and Prevention System using Snort . Abstract : System :," vol. 1, no. 5, pp. 1–6, 2012.

[5] S. Kumar, "Survey of Current Network Intrusion Detection Techniques," *Citeseer*, pp. 1–18, 2007.

[6] Z. Chiba, N. Abghour, K. Moussaid, A. El Omri, and M. Rida, "A Survey of Intrusion Detection Systems for Cloud Computing Environment," 2014.

[7] K. Thongkanchorn, S. Ngamsuriyaroj, and V. Visoottiviseth, "Evaluation studies of three intrusion detection systems under various attacks and rule sets," *IEEE Reg. 10 Annu. Int. Conf. Proceedings/TENCON*, vol. 6, pp. 6–9, 2013.

[8] Suricata, "Suricata Rules." [Online]. Available: https://redmine.openinfosecfoundation.org/projects/suricata/wiki/Suricata_Rules.

[9] OSSEC Project Team, "OSSEC Architecture." [Online]. Available: https://ossec.github.io/docs/manual/ossec-architecture.html.

[10] OSSEC, "OSSEC Rules." [Online]. Available: http://ossec-docs.readthedocs.io/en/latest/syntax/head_rules.html.

[11] M. Pihelgas, "a Comparative Analysis of Open- Source Intrusion Detection," *Tallinn Univ. Technol.*, pp. 1–67, 2012.

[12] A. Alhomoud, R. Munir, J. Pagna, I. Awan, and A. Al-dhelaan, "Performance Evaluation Study of Intrusion Detection Systems," vol. 5, pp. 173–180, 2011.

[13] Miguel A. Calvo Moya, "ANALYSIS AND EVALUATION OF THE SNORT AND BRO NETWORK INTRUSION DETECTION," 2008.

[14] A. Anand, "An Overview on Intrusion Detection System and Types of Attacks It Can Detect Considering Different Protocols," *Int. J. Adv. Res. Comput. Sci. Softw. Eng.*, vol. 2, no. 8, pp. 94–98, 2012.

[15] R. Kumar, S. P. Lal, and A. Sharma, "Detecting Denial of Service Attacks in the Cloud," *Proc. - 2016 IEEE 14th Int. Conf. Dependable, Auton. Secur. Comput. DASC 2016, 2016 IEEE 14th Int. Conf. Pervasive Intell. Comput. PICom 2016, 2016 IEEE 2nd Int. Conf. Big Data Intell. Comput. DataCom 2016 2016 IEEE Cyber Sci. Technol. Congr. CyberSciTech 2016, DASC-PICom-DataCom-CyberSciTech 2016*, pp. 309–316, 2016.

[16] J. W. K. and R. V. Allen Oommen Joseph, "Cloud Security Mechanisms for Data Protection : A Survey Cloud Security Mechanisms for Data Protection : A Survey," no. February, 2016.

[17] A. Bakshi and B. Yogesh, "Securing cloud from DDOS attacks using intrusion detection system in virtual machine," *2nd Int. Conf. Commun. Softw. Networks, ICCSN 2010*, pp. 260–264, 2010

[18] B. Khadka, C. Withana, A. Alsadoon, and A. Elchouemi, "Distributed Denial of Service attack on Cloud : Detection and Prevention," vol. 4, no. September, pp. 210–215, 2015.

[19] B. Kitchenham and S. Charters, "Guidelines for performing Systematic Literature

Reviews in Software Engineering," *Engineering*, vol. 2, p. 1051, 2007.

[20] Y. Lanjuan, "Defense of DDoS attack for cloud computing," in *IEEE International Conference on Computer Science and Automation Engineering (CSAE)*, 2012, pp. 626–629.\

[21] C. Wohlin, *Experimentation in Sofware Engineering*, vol. 53, no. 9. 2013.