EVALUATION OF THREE INTRUSION DETECTION SYSTEMS UNDER VARIOUS ATTACKS

KITTIKHUN THONGKANCHORN

**A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE (COMPUTER SCIENCE)
FACULTY OF GRADUATE STUDIES
MAHIDOL UNIVERSITY
2012**

Thesis
entitled

EVALUATION OF THREE INTRUSION DETECTION SYSTEMS UNDER VARIOUS ATTACKS

……..……..…..……
Mr. Kittikhun Thongkanchorn
Candidate

……..……..…..……
Lect. Sudsanguan Ngamsuriyaroj, Ph.D.
Major advisor

……..……..…..……
Asst.Prof. Vasaka Visoottiviseth, Ph.D.
Co-advisor

……..……..…..……
Lect. Thitinan Tantidham, Ph.D.
Co-advisor

……..……..…..……
Prof. Banchong Mahaisavariya,
M.D., Dip Thai Board of Orthopedics
Dean
Faculty of Graduate Studies
Mahidol University

……..……..…..……
Lect. Sudsanguan Ngamsuriyaroj, Ph.D.
Program Director
Master of Science Program in
Computer Science
Faculty of Information and
Communication Technology
Mahidol University

Thesis
entitled

EVALUATION OF THREE INTRUSION DETECTION SYSTEMS UNDER VARIOUS ATTACKS

was submitted to the Faculty of Graduate Studies, Mahidol University
for the degree of Master of Science (Computer Science)
on
May 29, 2012

……..……..…..……
Mr. Kittikhun Thongkanchorn
Candidate

……..…..……
Lect. Panita Pongpaibool, Ph.D.
Chair

……..…..…….……
Lect. Sudsanguan Ngamsuriyaroj, Ph.D.
Member

……..…..……...…
Lect. Thitinan Tantidham, Ph.D.
Member

……..…..……..
Asst.Prof. Vasaka Visoottiviseth, Ph.D.
Member

……..…..……...…
Prof. Banchong Mahaisavariya,
M.D., Dip Thai Board of Orthopedics
Dean
Faculty of Graduate Studies
Mahidol University

……..…..……..…
Assoc.Prof. Jarernsri L. Mitrpanont, Ph.D.
Dean
Faculty of Information and
Communication Technology
Mahidol University

ACKNOWLEDGEMENTS

This thesis would not have been possible without the guidance and support of several individuals who in one way or another contributed and extended their valuable assistance in the preparation and the completion of this study.

First of all, I offer my sincerest gratitude to my advisor, Dr. Sudsanguan Ngamsuriyaroj, who has supported me throughout my thesis with her guidance, caring, patience, and thoughtful discussions. I sincerely thank Dr. Vasaka Visoottiviseth for guiding my research and Dr. Thitinan Tantidham for the examination of this thesis. I am pleased by their close insight, their constructive and detailed feedbacks and their interest in my work. Special thanks also go to Dr. Panita Pongpaibool, who had participated in my final defense.

I would like to thank Lect. Pagaporn Pengsart who allows me to work in my own way and has guided me for every difficulty professionally and personally. In addition, I would like to thank all Ph.D. students who gave me valuable discussions and comments. I am also thankful to many professors and staff of the Faculty of Information and Communication Technology, Mahidol University, who had supported me during my academic years. I also give a special thank to the office of graduate studies that has given the most helps.

Finally, I would like to thank my family for their personal support and great patience at all times.

Kittikhun Thongkanchorn

EVALUATION OF THREE INTRUSION DETECTION SYSTEMS UNDER VARIOUS ATTACKS

KITTIKHUN THONGKANCHORN 5037735 ITCS/M

M.Sc. (COMPUTER SCIENCE)

THESIS ADVISORY COMMITTEE: SUDSANGUAN NGAMSURIYAROJ,Ph.D., VASAKA VISOOTTIVISETH, Ph.D., THITINAN TANTIDHAM,Ph.D.

ABSTRACT

Intrusion detection system (IDS) tools for detecting malicious traffic have been widely used in many organizations and they use a variety of technologies. Each IDS tool deploys different approaches and has been developed under different purposes. Intrusion detection systems include a set of IDS rules which can be defined differently. Therefore, choosing an IDS tool to work efficiently and appropriately in a specific environment would not be easy.

The goal of this research was to evaluate three popular open-source IDS tools in terms of performance and accuracy. The selected IDS tools were Snort, Bro and Suricata. In addition, their system architecture and the main components were compared and analyzed. The experiments conducted used various attack types including DoS attack, DNS attack, FTP attack, Scan port attack, and SNMP attack. Each experiment was run under different traffic rates and only a specific set of rules was active. Moreover, the performance metrics used to measure was the number of packets lost, the number of alerts, the CPU utilization and the memory usage. The results showed that each attack type had significant effects on the performance of each IDS tool in different ways. Specifically, Bro IDS showed better performance than other IDS tools when evaluated under different attack types and using a specific set of rules. The results also indicated that the accuracy dropped when the three IDS tools activate the full rule sets instead of a specific set of rules.

KEY WORDS: INTRUSION DETECTION SYSTEMS / SNORT / BRO / SURICATA / PERFORMANCE EVALUATION

88 pages

การประเมินระบบการตรวจจับการบุกรุกสามระบบภายใต้การโจมตีหลายแบบ
EVALUATION OF THREE INTRUSION DETECTIO SYSTEMS UNDER VARIOUS ATTACKS

กิตติคุณ ทองกัญชร 5037735 ITCS/M

วท.ม. (วิทยาการคอมพิวเตอร์)

คณะกรรมการที่ปรึกษาวิทยานิพนธ์: สุดสงวน งามสุริยโรจน์, Ph.D., วัสกา วิสุทธิวิเศษ, Ph.D.,ฐิตินันท์ ตันติธรรม,Ph.D.

บทคัดย่อ

ระบบการตรวจจับการบุกรุก (Intrusion Detection System, IDS) เป็นระบบใช้ตรวจจับความผิดปกติของข้อมูลในระบบคอมพิวเตอร์ ที่มีการใช้งานอย่างแพร่หลายในหลายหน่วยงานและมีเทคโนโลยีการทำงานหลายรูปแบบ IDSได้มีการพัฒนาเพื่อใช้งานในแบบต่างๆ โดยมีการกำหนดกฎหรือกติกาในการตรวจจับความผิดปกติ ดังนั้นการเลือกระบบที่เหมาะสมกับสิ่งแวดล้อมเฉพาะจึงไม่ใช่สิ่งที่ง่าย

จุดมุ่งหมายของงานวิจัยนี้เพื่อประเมินระบบตรวจสอบการบุกรุกที่เป็นระบบเปิด (Open-Source) และที่นิยมใช้งานอย่างแพร่หลายจำนวนสามระบบ คือSnort Bro และ Suricataโดยนำโครงสร้างของระบบและส่วนประกอบสำคัญมาทำการวิเคราะห์และเปรียบเทียบด้วยซึ่งในการทดลองได้มีการประเมินจากการสร้างสถานการณ์การโจมตีที่หลากหลาย เช่น DoS attack, DNS attack , FTP attack, SCAN port attack และ SNMP attack และแต่ละการทดลองได้มีการกำหนดความเร็วข้อมูลการจราจรที่แตกต่างกัน รวมทั้งมีการกำหนดกฎที่ใช้งานโดยเฉพาะไม่เหมือนกันส่วนตัวแปรที่ใช้วัดประสิทธิภาพในการทดลองคือ จำนวน Packet Loss จำนวน Alert ปริมาณการใช้งาน CPU และการใช้งาน Memory ผลการทดลอง แสดงให้เห็นว่าการโจมตีแต่ละชนิดได้ส่งผลอย่างมากและแตกต่างต่อประสิทธิภาพการทำงานของ IDS แต่ละชนิดโดย Bro มีความสามารถและมีความแม่นยำในการตรวจจับความผิดปกติของข้อมูลที่มีการโจมตีหลากหลายกว่า IDS ระบบอื่น นอกจากนี้ผลการทดลองได้แสดงว่าการใช้กฎทั้งหมดได้ลดความแม่นยำของการตรวจจับเมื่อเทียบกับใช้กฎเฉพาะในแต่ละการโจมตีของทุก IDS

88หน้า

CONTENTS

Page

CONTENTS (cont.)

CONTENTS (cont.)

LIST OF TABLES

LIST OF FIGURES

Figure **Page**

LIST OF FIGURES (cont.)

LIST OF FIGURES (cont.)

Figure **Page**

CHAPTER I
INTRODUCTION

The increasing of network traffic and threats has challenged the network security community for years. Threats are methods, tools, or techniques that intruders used to exploit vulnerabilities in several systems to achieve their goals. Currently, an intrusion detection system (IDS) is one of the efficient methods widely used to protect a network from intruders.

Staniford-Chen et al. (1998)[4], published the common intrusion detection framework which consisted of four main parts. The first was “sensors” that monitored applications, operating systems, or network activities. The second was “analyzers” which analyzed and detected the evidence of an intrusion from data outputs of the sensors. The third was “incidence storage” which collected and stored evidence and other information in the system. The last was “response units” which were humans or automated processes to carry out a set of actions controlled by the analyzer to respond when an intrusion was detected.

IDS cannot fully protect the system against attackers. However, it can present the useful information for the site administrator. Moreover, it provides an efficient way to protect the system from a malicious agent if it is correctly deployed as a complement to other security technologies.

According to the source of the traffic analyzed IDS is grouped into two categories: host-based and network-based. A host-based IDS monitors the host where the sensor is installed. A network-based IDS is placed on a network segment so that it can monitor the whole traffic directed towards one or more computers. Table 1.1 presents the comparison between the host-based and the network-based IDS.

Table 1.1 Comparisons of Host-based and Network-based IDS

Host-based IDS	
Advantages	Disadvantages
• Able to analyze each application • Able to verify the success of an attack • Able to detect attacks not involve the network • No additional hardware is required	• Must be installed on every single host • Degrade the system performance
Network-based IDS	
Advantages	Disadvantages
• No effect to the host performance • Able to monitor multiple hosts at the same time • Able to detect network attacks that are not visible from single hosts	• Need to handle huge information • Must be very fast to avoid packet lost • Difficult to deploy and configure • Difficult for encrypted channels

An IDS can be further classified as a signature-based or anomaly-based via the analysis techniques used to detect intrusions. A signature-based IDS creates a set of models to describe intrusive behaviors. Once these models have been written, they can be matched against the event stream to distinguish normal from malicious events. On the contrary, an anomaly-based system defines the normal behavior of the system and flags intrusive events that fall outside normal behaviors, or when there are some sufficient differences based on statistical perspectives. The comparison between the signature-based and the anomaly-based IDS is shown in Table 1.2.

1.1Motivation

The IDS tools used to detect network attacks have been widely used in many organizations and there are many varieties. Several IDS tools are signature-based and they are appeared as commercial, open-source, or even research tools. The well-known IDS include Snort [20], Bro [3], Suricata [23], OSSEC [14], and Prelude [16]. Snort created by Martin Roesch is the most popular open source installed IDS. The main feature of Snort is the rule-based engine that performs the content pattern matching and detects a variety of attacks and probes. The detection engine is usually programmed to perform per packet tests and actions.

Table 1.2 Comparisons of Signature-based and Anomaly-based IDS

Signature –based IDS	
Advantages	Disadvantages
• Higher detection rate and less false positives • Do not require a learning phase • Can provide more information on the attacks	• Need signatures update • Cannot detect unknown attack
Anomaly-based IDS	
Advantages	Disadvantages
• Do not require continuous maintenance • The learning algorithm can adapt the system to the operating environment • Can detect unknown attacks	• Prone to false positives • Do not provide attack identification

Bro [3] is the open source IDS software widely used in research to verify results of other IDS. The outstanding features of Bro are the domain-specific scripting language that enables site-specific monitoring policies, and the ability to perform on high-performance networks. It is typically operated at a variety of large sites, and can analyze many protocols.

Suricata [23] is a new generation of the open source IDS engine developed recently in the past few years by Open Information Security Foundation (OISF). Suricata like Snort supports the rules from VRT and emerging threats. The main features of Suricata are multi-threading and automatic protocol detection engine.

OSSEC [14] is an open source host-based IDS. It performs log analysis, file integrity checking, policy monitoring, rootkit detection, real-time alerting and active incident response.

Prelude [16] is a universal Security Information Management (SIM) system. It collects, normalizes, sorts, aggregates, correlates and reports all security-related events independently of product brands or licenses giving rise to such events. In other words, Prelude is agent-less. In addition to be capable of recovering all logs (system logs, syslog, and flat files), Prelude receives the benefits from the native support having a number of systems dedicated to enriching information.

Many researchers [1, 8, 11, 12, and 21] studied IDS in terms of performance or accuracy in order to reduce the number of alarms in both false positives and false negatives. However, IDS sensors produce different alarms, and different attack description has been reported. In addition, each IDS has different approaches, features and strengths. Therefore, this research aims to evaluate the performance and the accuracy of the three popular open source IDS tools: Snort, Bro and Suricata.

1.2 Problem Statement

To choose an IDS tool to work appropriately with a specific environment is a challenging task because IDS tools are implemented using different approaches and they are developed with different purposes. As a result, the techniques of intrusion detection and IDS rules will be defined differently. Moreover, the IDS outputs would give individual attack description differently. This study will evaluate the selected three IDS in several perspectives.

1.3 Objectives of the Research

The objectives of the research are as follows:

1. To analyze and compare the three IDS software in terms of system architecture, and their main components including the detection engine, the rule syntax, and the reported outputs.

2. To evaluate the three IDS software using a specific set of rules for single attack types such as probe and scan vulnerabilities, the initial access, and the privilege escalation.

3. To generate a variety of single traffic protocols including TCP, UDP, FTP, and HTTP as well as the mixed traffic for use during the evaluation.

4. To measure the accuracy based on the value of false positives and false negatives of the outputs or the alerts.

5. To measure the IDS performance based on the number of packet lost.

1.4 Scope of the research

The scope of this research is described as follows:

1. To evaluate the performance and the accuracy of the three IDS software: Snort Bro and Suricata.

2. To apply a specific set of shared rules for each IDS.

3. To generate both the attack traffic and the normal traffic for evaluation.

1.5 Organization of the thesis

Chapter I: Introduction. This chapter describes the motivation, the problem statement, the objectives and the scope of the thesis.

Chapter II: Background. This chapter explains the concept and background of intrusion detection software including the Common Intrusion Detection Framework (CIDF) and the architecture of the three IDS: Snort Bro and Suricata.

Chapter III: Literature Review. This chapter presents the related work and the review of research in this area.

Chapter IV: Experimental Results. This chapter gives the details of the experiments conducted as well as the results obtained.

Chapter V: Conclusions. This chapter discusses and concludes our work.

CHAPTER II
BACKGROUND

This chapter describes the concept of the Common Intrusion Detection Framework (CIDF). The features and the architecture of the three selected IDS: Snort, Bro and Suricata, are also presented. In addition, the software used to generate the traffic in this research is explained.

2.1 Common Intrusion Detection Framework (CIDF)

The CIDF [19] as shown in Figure 2.1 is a commonly published framework that describes an intrusion detection system. The CIDF divide the IDS software into four parts: sensors, analyzers, incident storage, and response units.

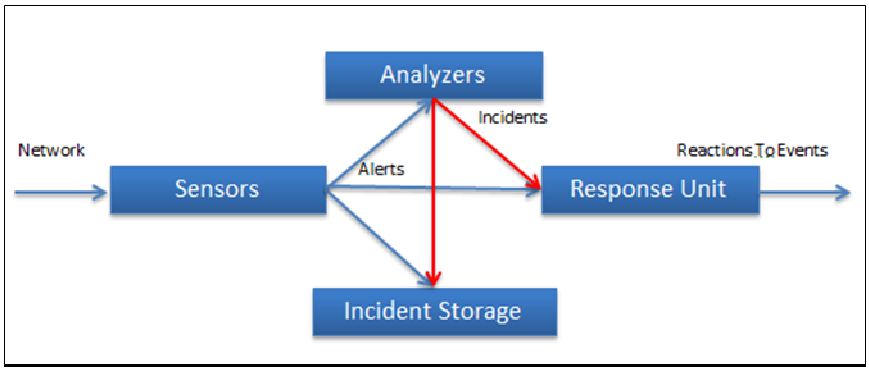


Figure 2.1 Components of the CIDF

Sensors were often referred to as information sources or event generators. They obtained information from the large computational environment outside the intrusion detection system. The most common sources in intrusion detection were networks, hosts, and applications. Typically, these sensors generate a lot of alerts, and all of them would be analyzed. However, only the relevant alerts were reported as incidents.

To decide whether those events indicated signs of intrusions, the analyzers used the outputs of the sensors to model and analyze the collected data events. Today, intelligent techniques have become indispensable tools for the IDS analyzers. Moreover, there are two types of analyzers: pattern-based and anomaly-based.

The incident storage stored all alerts to support the analysis process which the instances of alerts are used to classify an alert. The databases are not expected to change or process the alerts in any way.

The response units could be humans or automated processes which carry out a prescription controlled by the analyzer. The analyzer instructs them to act when an intrusion is detected. The actions can be passive measures such as reporting intrusion alerts to administrators, or active measures such as blocking intrusions.

2.2 Snort

Snort [20] is a free and open source network intrusion prevention system (IPS) and network intrusion detection system capable of performing packet logging and real-time traffic analysis on IP networks. Snort was originally written by Martin Roesch and is now developed by Sourcefire, of which Roesch is the founder and the chief technology officer. Snort's integrated enterprise versions with specific built hardware and commercial support services are sold by Sourcefire.

Snort performs protocol analysis, and content searching/matching. It is commonly used to actively block or passively detect a variety of attacks and probes, such as buffer overflows, stealth port scans, web application attacks, SMB probes, and OS fingerprinting attempts. Snort has been mostly used for intrusion prevention purposes, by dropping attacks as they are taking place. It can be combined with other software such as Snort, OSSIM, and the Basic Analysis and Security Engine (BASE) to provide a visual representation of intrusion data. It is the most widely deployed intrusion detection and prevention technology worldwide. It also has the most numerous and active community in the open source IDS software today. In addition, Snort uses different licensing for the engine and the rule-sets.

2.2.1 Snort Architecture

Snort architecture as shown in Figure 2.2 contains six main components described below.

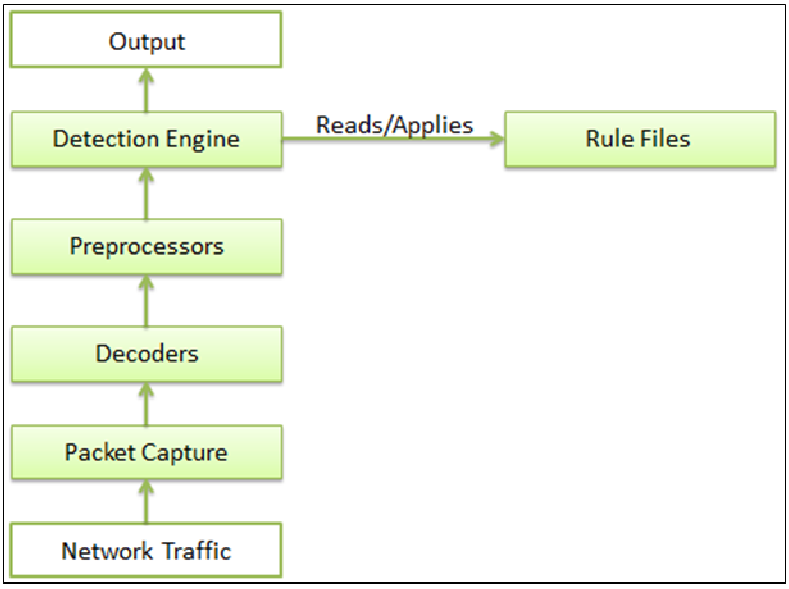


Figure 2.2 Snort Architecture

1. Decoders

Decoders fit the captured packets into data structures and identify the relevant link level protocols. Later, it decodes the packets' IP, and TCP or UDP in order to get other useful information like ports and addresses. Snort will issue an alert if it finds malformed headers, or unusual length TCP options.

2. Preprocessors

Preprocessors are another filter used to identify things that would be examined later by the next modules such as the detection engine. The events include suspicious connection attempts to some TCP/UDP ports or too many UDP packets sent in a short period of time (called a port scan attack). The preprocessors function is to examine packets which are potentially dangerous for the detection engine in an effort to find known patterns.

3. Rules Files

Rules Files are the plane text files containing a list of rules with a known syntax. The syntax includes protocols, addresses, and the output plug-ins associated. Those rules files can be updated similar to the way the virus definition files are updated.

4. Detection Plug-ins

Detection Plug-ins modules are referenced from the definition in the rules files, and they are intended to identify patterns whenever a rule is evaluated.

5. Detection Engine

Making use of the detection plug-ins, it matches packets against rules previously loaded into memory since the initialization.

6. Output Plug-ins

These modules allow the notifications such as the alerts and the logs to be accessed by users in many ways such as the monitor, the files, or the databases.

2.2.2 Snort Rule Structure

Snort rule structure as shown in Figure 2.3 consists of the rule headers containing the necessary information that every rule must have, and the rule options containing a list of optional information used to refine a match. One Snort's sample rule is shown in Figure 2.4.

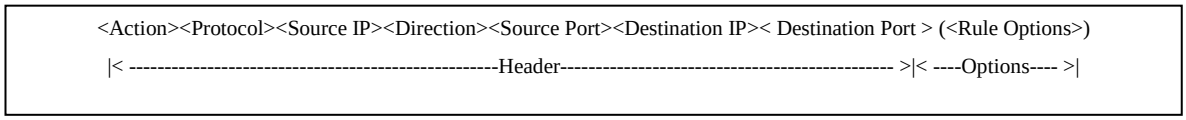


Figure 2.3 Snort Rule Structure

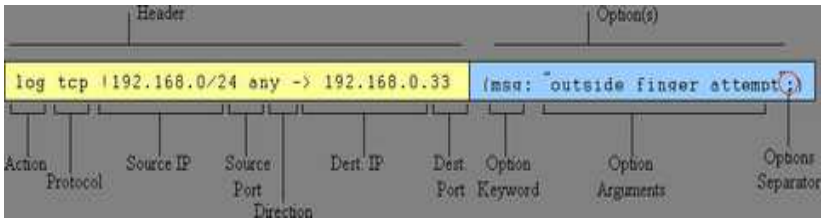


Figure 2.4 Snort Rule Example

The rule headers consist of five components briefly described as follows:

1. *Action:* Action tells Snort what to do when a match occurs. A common action is to log information to an alert file. Five available actions in Snort are alert, log, pass, activate, and dynamic. Moreover, for Snort in the inline mode, additional options include drop, reject, and sdrop.

2. *Protocol:* The following protocols: TCP, UDP, ICMP or IP are specified.

3. IP source and IP destination: The IP of source or destination is defined as a single IP address, the CIDR (Classless Inter-Domain Routing) block, a range of IP addresses, or the keyword “any” for all possible IP addresses.

4. *Port source and Port destination:* The port number of the source or destination can be specified as a single static port, or the keyword “any” for any port number.

5. *Direction:* The direction operator,→, indicates the orientation or the direction of the traffic that the rule is applied.

The rule options consist of five components briefly explained as follows:

1. *Rule options:* The rule options provide the detail of matching parameters and bind a rule to a rule identification system.

2. *General:* These options provide information about the rule such as reference information, rule identification, and specific log messages.

3. *Payload*: These options examine data contained in a network packet such as content matching expressions.

4. *Non-Payload:* These options provide matching specifications against packet header data beside port numbers and IP addresses.

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

2.3 Bro

Bro [3] is a well known intrusion detection system developed by Vern Paxson Bro is designed to achieve high-speed monitoring, real-time notification and extensible IDS. By monitoring the high-speed network and the large volume data flows, Bro does not tolerate to drop any packet. Notifications have to be issued as fast as possible when it detects the attempted attacks. It is essential to make the system grow strong, customizable, and extensible. The policies are written in Bro language which is extensible and flexible. Furthermore, Bro does assume that the monitor will be attacked, as well as that attackers are familiar with Bro's system structure and design Thus, it must be get ready for the attacks. Thus, the design of Bro is very challenging.

2.3.1 Bro Architecture

Bro Architecture has the essential modules as shown in Figure 2.5:

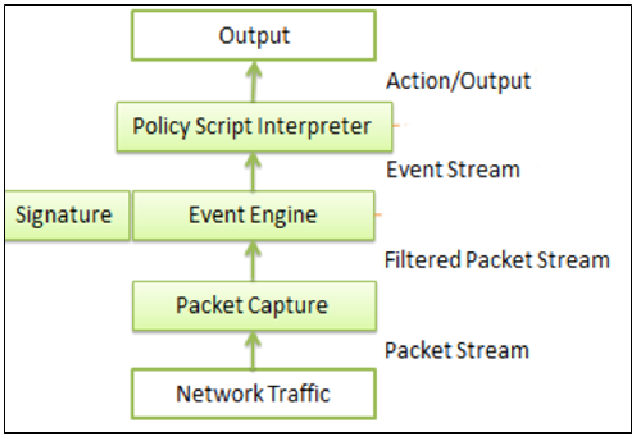


Figure 2.5 Bro Architecture

- *Packet Capture:* Like Snort, Bro captures traffic using libpcap which helps in porting Bro to different UNIX flavors, and allows it to operate

on tcpdump packet traces as the offline analyzer. Using libpcap also brings another advantage as it allows basic packet filtering which is very useful in environments where only certain traffic must be analyzed.

- *Event Engine:* This layer performs several integrity checks to assure that the packet headers are well-formed. For example, it verifies whether the IP header checksum is correct. At this stage, Bro reassembles IP fragments so the network layer analyzer can complete IP datagrams. It also sends the events to the Policy layer.
- *Signature Engine:* It inspects the packet stream, and generates an event each time a signature is matched. Such events will be further analyzed by a policy script.
- *Policy Layer:* The policy script interpreter executes scripts written in a specialized Bro language. The scripts specify event handlers which execute arbitrary Bro scripting commands to generate new events, to log real-time notifications, and to record data to disk.

2.3.2 Bro-IDS Rule Structure

Bro provides a different language specifically designed to define pattern-matching rules called signatures as shown in Figure 2.6. These signatures are very similar to the Snort rules, and most of them can be translated from the Snort rules. When the signatures are matched Bro and Snort would generate special events which can be analyzed by the policy scripts.

In addition, Bro is designed to provide extra context information to its signature. The policy script can test the software running on the target machine or consider in which way the server replies to the malicious traffic to understand whether an attack succeeded or not.

signature id { attribute-set }

Figure 2.6 Bro Signature

The signature id is defined uniquely for each signature. There are two types of attributes: conditions and actions. The conditions define when the signature is matched, and they have four types: header, content, dependency, and context. The actions declare what to do in the case of a match. Figure 2.7 shows an example of a Bro signature.

2.4 Suricata

Suricata [23] is open source intrusion detection and prevention engine which is not intended to just replace or emulate the existing tools in the industry, but to introduce new ideas and technologies. The Suricata engine and the HTTP Library are available to use under the GPLv2. Suricata is a rule-based engine that utilizes externally developed rule sets to monitor network traffic and also provides alerts to the system administrator when suspicious events occur. It is designed to be compatible with existing network security components. Thus, Suricata features would unify the output functionality and pluggable library options to accept calls from other applications.

```
signature formmail-cve-1999-0172 {
ip-proto == tcp
dst-ip == 1.2.0.0/16
dst-port = 80
http /.*formmail.*\?.*recipient=[^&]*[;|]/
event "formmail shell command"
 }
```

Figure 2.7 Example of Bro Signature

2.4.1 Suricata Architecture

Suricata has not yet published its system architecture model. However, the system can be described using packet pipeline and Suricata module. The pipeline also uses *pcap* to capture packets from the network traffic. Figure 2.8 shows the Suricata packet pipeline.

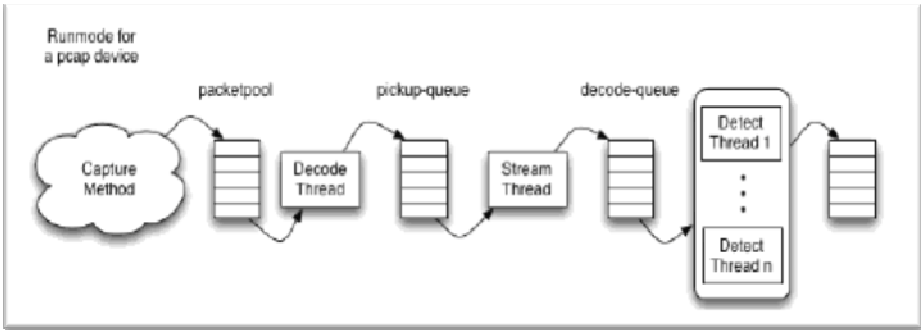


Figure 2.8 Suricata Packet Pipeline

Modules are used to encapsulate a single primary function with lifecycle call backs. Each thread in the packet pipeline is an instance of a module. These threads are initialized by the run mode. The run mode also initializes the queues and packet handlers used for moving packets between modules and queues. A thread is marked as runnable after all the steps from the run mode initialization are complete.

- *Capture Module*: The pcap device is initialized using the name provided, for example,"eth0". Once the device is initialized, it will begin gathering packets and passing them to the decoder. Suricata would act as a thin wrapper around the data provided, making it compatible with the link type decoders.
- *Decode Module*: Decoding is the process of taking a buffer and converting its contents to the Suricata support data structure. These buffers are handed off to a specific link type decoder.
- *Detection Module*: The detection module takes care of multiple complex tasks including loading all signatures, initializing detection plugins, creating detection groups for packet routing, and finally running packets through all applicable rules.

2.4.2 Suricata rule Structure

The most Suricata signature used is from emerging threats. Emerging threats pro and source fire's VRT are the same as on the Snort, but the Suricata rule is separated into three parts: action, header, and options, as shown in Figure 2.9. The example of Suricata signature is shown in Figure 2.10.The three parts of a Suricata rule are described below.

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

Figure 2.9 Suricata Rule Structure

1. ***Action:*** It determines what will happen when a signature matches. Four types of actions are Pass, Drop, Reject, and Alert. The rules are loaded in the order of the important so that the most important signatures will be scanned first. The default order is: pass, drop, reject, alert. But, the order of priority could be changed. Those four actions are briefly explained as follows.

```
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 2.10 Suricata Rule Example

- *Pass:* If a signature matches and contains “Pass”, Suricata stops scanning the packet and skips to the end of all rules.
- *Drop:* This action concerns only the inline mode. If the program finds a signature that matches, and contains “Drop”, it stops immediately, and the packet will not be sent any further.
- *Reject:* This action is an active rejection of the packet. Both receiver and sender receive a reject packet. There are two types of reject packets that will be automatically selected. If the offending packet concerns TCP, it will be a Reset-packet. For all other protocols, it will be an ICMP-error packet. Suricata also generates an alert. When it is in the inline mode, the offending packet will also be dropped.
- *Alert:* If a signature matches and contains “Alert”, the packet will be treated like any other non-threatening packet, except that an alert will be generated by Suricata.

2. ***Header:*** The header must contain four types of information: Protocol, IP, Port, and Direction.

3. ***Rule Options***: It provides the detail of matching parameters and binds a rule to a rule identification system. Many options are available for selection as given below.

- msg (message) gives more information about the signature and the possible alert;
- rev (Revision) represents the version of the signature;

- sid (Signature ID) and gid (Group ID) give every signature and group its own id, respectively;
- Class Type gives the information about the classification of rules and alerts;
- Reference keywords direct to places where the information about the signature and the problem the signature tries to address.

2.5 Summary

Table 2.1 compares the features of the three IDS. For multithread processing, Bro has a fully functional cluster deployment model which helps users to scale support on a single box and/or across multiple boxes.

Table 2.1 Comparison of Three IDS Features

Features	**Bro v.1**.5	**Snort v.2.9**	**Suricata v.1.1**
Multi-Threaded Processing	No	No	Yes
Complete IPv6 Support	Yes	Some	Yes
IP Reputation	Some	No	Yes (soon)
Automated Protocol Detection	Yes	No	Yes
GPU Acceleration	No	No	Yes (soon)
Global Variables/Flowbits	Yes	No	Yes (soon)
Inline Windows Support	No	No	Yes
GeoIP Lookups	Yes	No	Yes (soon)
Advanced HTTP Parsing	Yes	No	Yes
HTTP Access Logging	Yes	No	Yes
SMB Access Logging	No	No	Yes (soon)
HTTP Blocklist Lookups	Yes	No	Yes (soon)
Free	Yes	Some	Yes

Table 2.2 shows a rule for detecting the scan attack. The signature of Snort and Suricata provided by the Emerging Threats official rule and the Bro signature provided by Bro IDS official site as they are converted from Snort rules.

Table 2.2 Comparison of Three IDS Signature of the Same Action

Bro	signature sid-1638 {
	ip-proto == tcp
	src-ip != local_nets
	dst-ip == local_nets
	dst-port == 22
	event "SCAN SSH Version map attempt"
	tcp-state established,originator
	payload /.*[vV][eE][rR][sS][iI][oO][nN]_[mM][aA][pP][pP][eE][rR]/
	}

Snort	alerttcp $EXTERNAL_NET any -> $HOME_NET 22 (msg:"SCAN SSH Version map attempt"; flow:to_server,established; content:"Version_Mapper"; fast_pattern:only; classtype:network-scan; sid:1638; rev:6;)
Suricata	alerttcp $EXTERNAL_NET any -> $HOME_NET 22 (msg:"SCAN SSH Version map attempt"; flow:to_server,established; content:"Version_Mapper"; fast_pattern:only; classtype:network-scan; sid:1638; rev:6;)

The signature of IDS software can be classified into groups based on the type of the signature having the same behaviour. They have nine groups given in Table 2.3.

Table 2.3 Examples of Signatures Types

Signature Type	Snort	Suricata	Bro
dos	dos	dos	dos
	ddos	ddos	ddos
scan	scan	scan	scan
finger	finger	finger	finger
ftp	ftp	ftp	ftp
	tftp	tftp	tftp
telnet	telnet	telnet	telnet
dns	dns	dns	dns
icmp	icmp	icmp	icmp
mail	smtp	smtp	smtp
	pop3	pop3	pop3
backdoor	backdoor	backdoor	backdoor

In addition, they can be separated into two groups. Snort and Suricata use the same type of signatures whereas Bro uses its own language to write a signature. However, Bro has a feature to support Snort and Suricata's rule format, and its name is Snort2Bro. This feature can convert the signature, but not all rule context scan be converted. In fact, the names and the descriptions of contexts are different. The signature component of the three IDS has a few differences, but they contain the same main parts such as ID address and options. Thus, in our work, we evaluate the three IDS software using the same type of rules to compare their accuracy and performance. As a result, we can use such information to choose the appropriate IDS for a specific environment.

The three IDS engines mostly share the common supporting software requirements which are dependent on the configuration options. Table 2.4 shows the required libraries and software packages. Table 2.5 shows the useful libraries and software packages options required that should be considered when installed.

Table 2.4 Required Libraries and Software Packages of Three IDS

Package	**Description**	**Required by IDS**
Autotools	The “autotools” consist of autoconf, automake, and libtool. These will likely be installed on your system. You need the autotools if you will be using source from the Bro’s Subversion repository. You will need to run autogen.sh after you check out the code. We will go through the steps below.	Bro, Suricata
BIND8 headers and libraries	Most OSs will have BIND installed by default. BIND (Berkeley Internet Name Domain) is an implementation of the Domain Name System (DNS) protocols.	Bro
Bison or byacc	Most OSs will have bison installed by default. Bison is a general-purpose parser generator that converts an annotated context-free grammar into an LALR (1) or GLR parser for that grammar.	Bro, Suricata
Flex	Most OSs will have flex installed by default. Flex is a tool for generating scanners. A scanner, sometimes called a tokenizer, is a program which recognizes lexical patterns in text.	Bro, Suricata
Libdnet	Libdnet provides a simplified, portable interface to several low-level networking routines.	Snort
Libpcap	Most OSs will have libpcap installed by default. It is the packet capture library. You may need to install it with support large files (files large than 2G). If you have a Linux kernnel, you will want to configure libpcap for PF_RING support.	Bro, Snort, Suricata
LibYAML	LibYAML is a YAML parser and emitter written in C that is used to parse the configuration file.	Suricata
PCRE	The PCRE library is a set of functions that implement regular expression pattern matching using the same syntax and semantics as Perl 5.	Snort

Table 2.5 Useful Libraries and Software Packages Options

Package	**Description**	**Optional Required**
GnuPG	The OpenPGP standard.	Bro, Suricata
libcap-ng	The libcap-ng library is intended to make programming with POSIX capabilities much easier than the traditional libcap library.	Suricata
LibGeoIP	Ability to determine the location of IP addresses.	Bro, Suricata
Libmagic	Add ability to determine file types, as with the ftp analyzer.	Bro
libnet	Libnet is a generic networking API that provides access to several protocols.	Suricata
libnetfilter_queue	libnetfilter_queue is a userspace library providing an API to packets that have been queued by the kernel packet filter.	Suricata
libnfnetlink	libnfnetlink is the low-level library for netfilter related kernel/userspace communication. It provides a generic messaging infrastructure for in-kernel netfilter subsystems (such as nfnetlink_log, nfnetlink_queue, nfnetlink_conntrack) and their respective users and/or management tools in userspace.	Suricata
OpenSSL	Tough to image a system not having OpenSSL installed. It is needed to analyzessh certificates by the HTTP analyzer and for encrypted Bro to Bro communication.	Bro
PF_RING	PF_RING is a new type of network socket that dramatically improves the packet capture speed.	Bro, Snort, Suricata
zLib	Libz is a compression library. It is used for decompressing HTTP bodies by the HTTP analyzer, and for compressed Bro-to-Bro communication.	Bro
XML Analyzer	The XML analyzer is highly-experimental code written by Tobias Kiesling.	Bro, Suricata

CHAPTER III
LITERATURE REVIEW

This chapter gives the reviews of five selected papers that are related to our work. They are listed below.

- Performance Evaluation Comparison of Snort NIDS under Linux and Windows Server [8]
- Investigation of the Intrusion Detection System "Snort" Performance [12]
- Measurement of Snort Performance under Various Attacks [21]
- Performance Evaluation Study of Intrusion Detection System [1]
- Analysis and Evaluation of the Snort and Bro Network Intrusion Detection Systems [11]

3.1 Performance Evaluation Comparison of Snort NIDS under Linux and Windows Server [8]

This paper evaluated the effect of the operating system parameters of any platform to the performance of the Snort IDS. The authors measured the Snort performance under different operating systems including Linux and Windows 2003 Server. They used the default configuration of the IDS from the official site [3, 20, 23]. Figure 3.1 shows the Linux and Windows kernel support architecture for Snort. In addition, the experiments are conducted to evaluate the Snort running on two operating systems based on various Snort parameters on different network traffic. For example,

- Linux: different values of NAPI (2, 3, 20, 300) for the packet reception mechanism.

- Windows: Different CPU scheduling time to kernel's networking subsystem or to user process.
- Network traffic: the normal traffic contains packets recognized by Snort as normal, and the malicious traffic contains packets recognized by Snort as malicious.

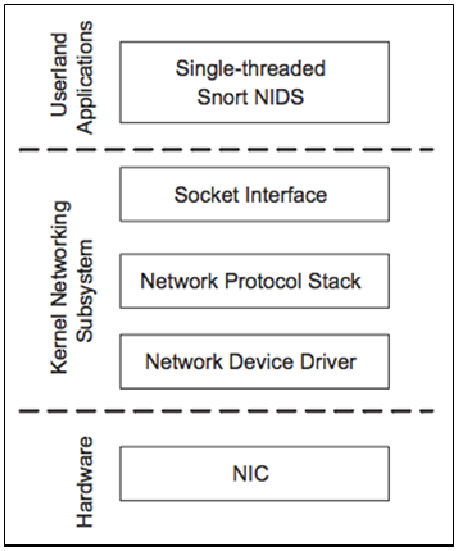


Figure 3.1 Linux and Windows Kernel Support Architecture for Snort [8]

Malicious traffic imposes high processing on Snort due to the triggering of events and logging. In order to generate malicious traffic, they had to modify and compile the traffic generator KUTE code to insert a string of ‘‘malicious.exe’’ at the end of the payload of the generated packets. To measure the performance of Snort under malicious traffic, a new rule is added to Snort's default rule-set as shown in Figure 3.2. The rule specifically checks every incoming UDP packet for a payload containing the string ‘‘malicious.exe’’. When a match occurs, a message of ‘‘Malicious packet has been detected’’ is outputted to the alert file stored in the default log directory with an identity of ‘‘44652’’. The exact format of the rule is as follows:

```
alertudp any any -> any any (content: “malicious.exe”; msg: “Malicious packet has been detected”; sid:44652;)
```

Figure 3.2 Sample Rule for Malicious Traffic

The testing environment as depicted in Figure 3.3 consisted of two machines: a sender and a receiver. The sender used KUTE to generate traffic packets for 30 seconds at a specific rate of 50 to 350 kpps. Two key performance metrics measured are Snort's average throughput and packet loss, where the computation is given in Figure 3.4.

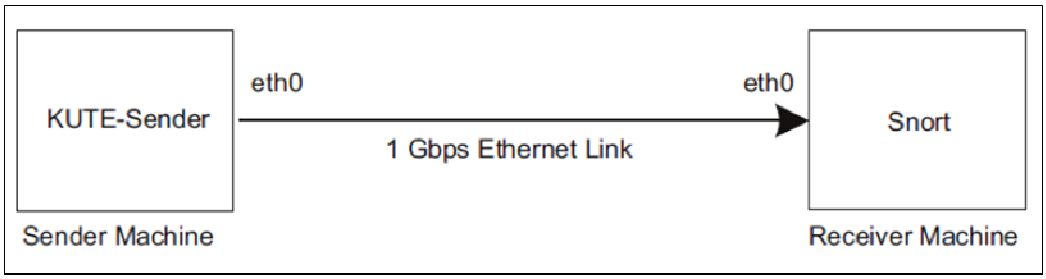


Figure 3.3 Testing Environment

$$\text{Snort's average throughput} = \text{the total number packets analyzed by Snort over 30 seconds}$$

$$\text{Snort's packet loss probability} = \frac{\text{The total number packets dropped by Snort}}{\text{Total packets received}}$$

Figure 3.4 Snort's Average Throughput and Packet Loss Probability

The results are plotted as four graphs shown in Figure 3.5. Each graph is given the brief description as follows.

a. Snort's throughput for both Linux and Windows for incoming normal traffic
b. Snort's throughput for both Linux and Windows for incoming malicious traffic
c. Snort's packet loss for incoming normal traffic
d. Snort's packet loss for incoming malicious traffic

When comparing the Snort performance under Linux and Windows, it was obvious that Linux with small NAPI values yields better performance over Windows, especially under malicious traffic.

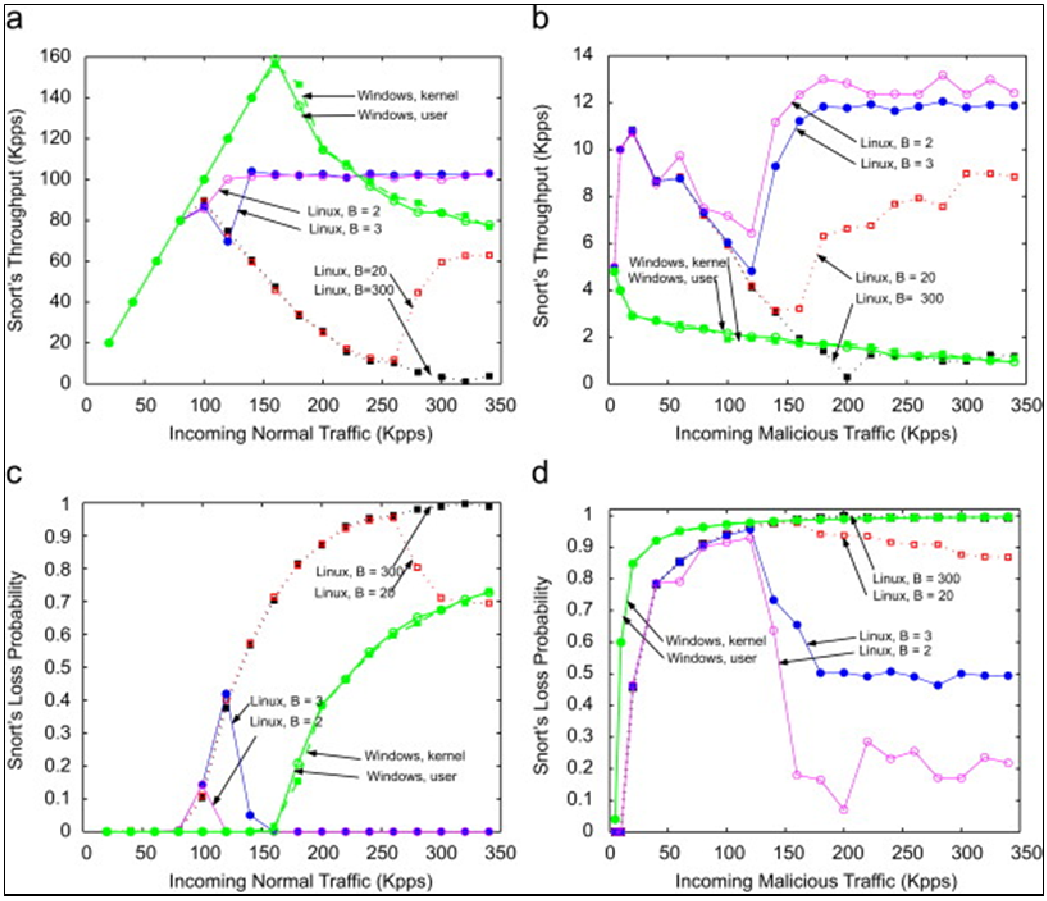


Figure 3.5 Results of Snort Performance on Incoming Normal and Malicious Traffic

3.2 Investigation of the Intrusion Detection System "Snort" Performance [12]

This paper considered three parameters: hardware, logging technique, and the pattern matching algorithm. We describe each parameter in turn below.

[1] *Hardware:* The main hardware components having significant effects to the performance of IDS are CPU, memory, the system bus, and the network interface card (NIC). Thus, different CPU and NIC are used and listed as follows.

- CPU: Pentium D940, Pentium IV, and Pentium III
- NIC: Marvell Yukon Gigabit Ethernet, Intel PRO/100, Intel PRO/1000, Realtek Fast Ethernet

[2] *Logging technique.* The data of intrusions were logged in two ways: (1) they are saved by Snort in MySQL database, and (2) they are saved in the file in the binary format and later used the Barnyard tool to send to Snort without having to spend time for SQL query, to send data to the database, to receive the query responses. Thus, there are two ways to log data.

- Snort Logging to MySQL, and it is the default of Snort 2.8.
- Snort Logging to the file using Barnyard tool sent logs to MySQL.

[3] *Pattern matching algorithm*. The packets were examined according to the rules that a new aho-corasick pattern-matching algorithm is applied in Snort version 2.6. The earlier version used the wu-manber algorithm. But, the Aho-corasick algorithm is faster although it used more RAM. According to the default setting, Snort 2.8 used the Aho-corasick algorithm. However, in the system with less memory, the lowmem algorithm would be chosen instead.

- Aho-corasick algorithm (default of Snort 2.8, fast but using more RAM)
- Lowmem algorithm (using less RAM)

The paper presented the investigation model as shown in Figure 3.6, and the model has two computers. One computer sent the network traffic and another computer had the Snort IDS installed. Figure 3.6 displayed how two logging techniques are used. The “Tcpdump” in the model was used to capture the real network traffic and Nessus is used to generate extra malicious network packets. The “Tcpreplay” was used to choose the traffic replay rate. Thus, how the IDS processed the same network packets sent at a different rate was investigated.

The results of Snort performance on different network traffic cards were shown in Figure 3.7 where the number on the graphs indicated the following NICs: (1) Realtek NIC, (2) Realtek NIC and lowmem pattern matching algorithm, (3) Intel PRO/100, (4) Intel PRO/1000, and (5) Intel PRO/1000 with Barnyard for data logging. Figure 3.8 displayed the results of CPU usage for three CPU types: (1) Pentium III, (2) Pentium IV, and (3) Pentium D.

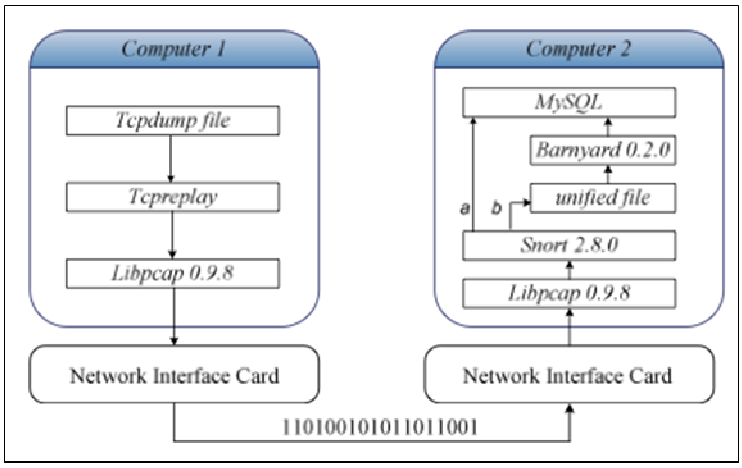


Figure 3.6 Investigation Scheme of Snort Performance

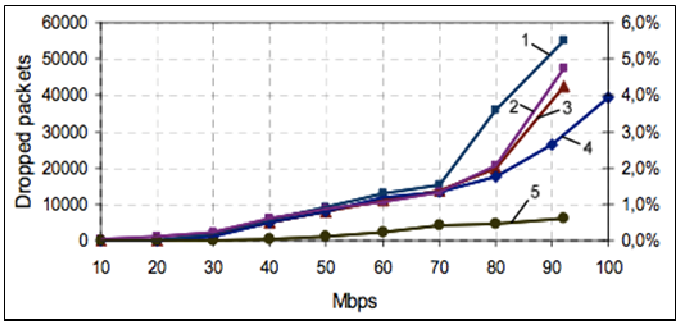


Figure 3.7 Results of Dropped Packets for Different NIC

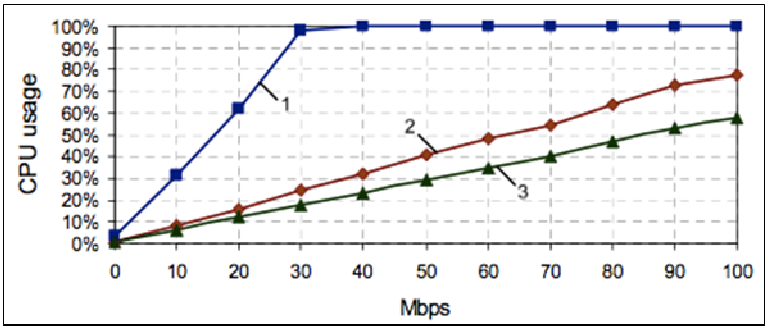


Figure 3.8 Results of CPU Usage for Different CPU

As a result, the hardware types and the alert logging techniques were the main factors that affect the Snort performance.

3.3 Measurement of Snort Performance under Various Attacks [21]

This paper measured the Snort performance under various attacks using different set of rules, either the full set of Snort rules or a specific set of rules. In addition, the number of attacking machines was varied from 1 to 15. The experiments considered two issues as given in Table 3.1. The first issue was the attacking tools which covered a variety of common protocols, and they are listed below.

- Ping: ICMP flooding, UDP flooding, SNMP Brute force, HTTP ping
- NMAP: scanning UDP and TCP ports

The second issue is the set of active Snort rules during each attack. Two performance metrics were measured. The first is the CPU utilization of the Snort process, and the top command was used to monitor and record it. The second is the amount of disk space used to keep logs, and the software called Cacti was used to do it.

Table 3.1 List of Attacking Tools and Rules Set

Experiment	Snort Rules Set
Ping-of-Death Attack	*ServiceImpacting*
HTML Attack	*Webserver-specific*
UDP Flooding	*Define our own rules*
Nmap –sU	*Low-level protocols*
Nmap –sX	*Scanning and Probing activities*
SNMP BruteForce	*Low-level protocols*

The experimental model having the system configuration shown in Figure 3.9 consisted of five machine groups as follows:

- The target servers to be attacked;
- The Snort machine for trapping and analyzing the network traffic. It also acted as the default gateway;
- A set of 15 attacking machines;
- Two data servers were used to collect all experimental results;
- Two client machines were used for monitoring the Snort performance.

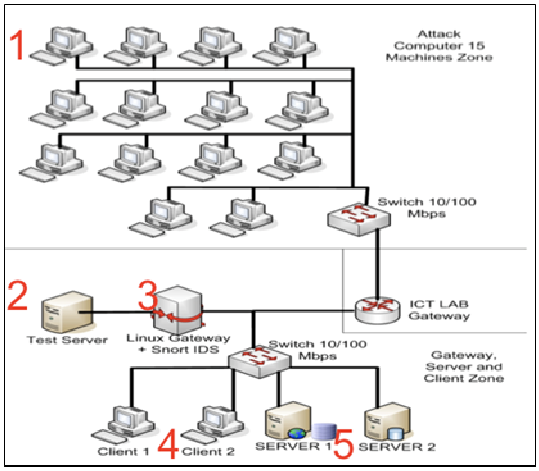


Figure 3.9 System Configuration

Figure 3.10depicted the average CPU usage of the Snort process for all six attacks when the full set of Snort rules is active. It can be seen that the CPU usage had been affected by the number of attacking machine in some attacks.

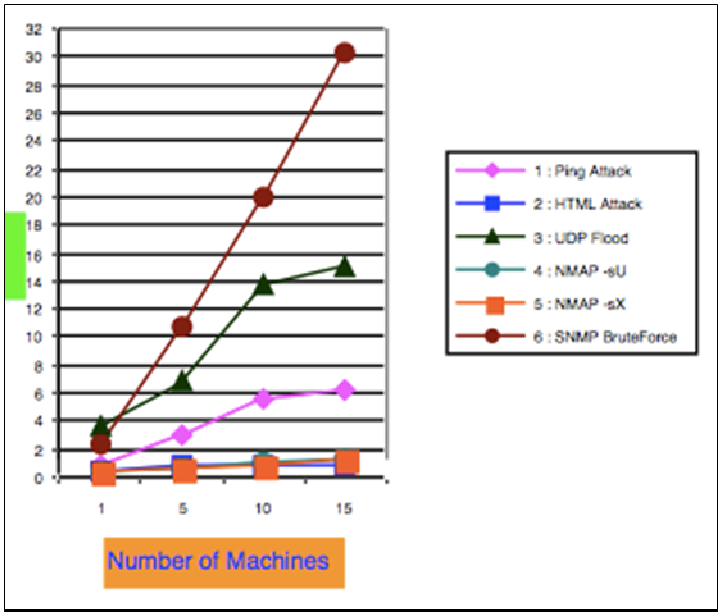


Figure 3.10 CPU Usage of Snort Process

Figure 3.11showed the experimental results when using two different rules sets for each of the six attacks: ping, nmap –sU, SNMP attack, HTMP, UDP flooding and nmap –sX. We can conclude that the number of active Snort rules did not affect every attack type since the number of alerts for each attack type was different.

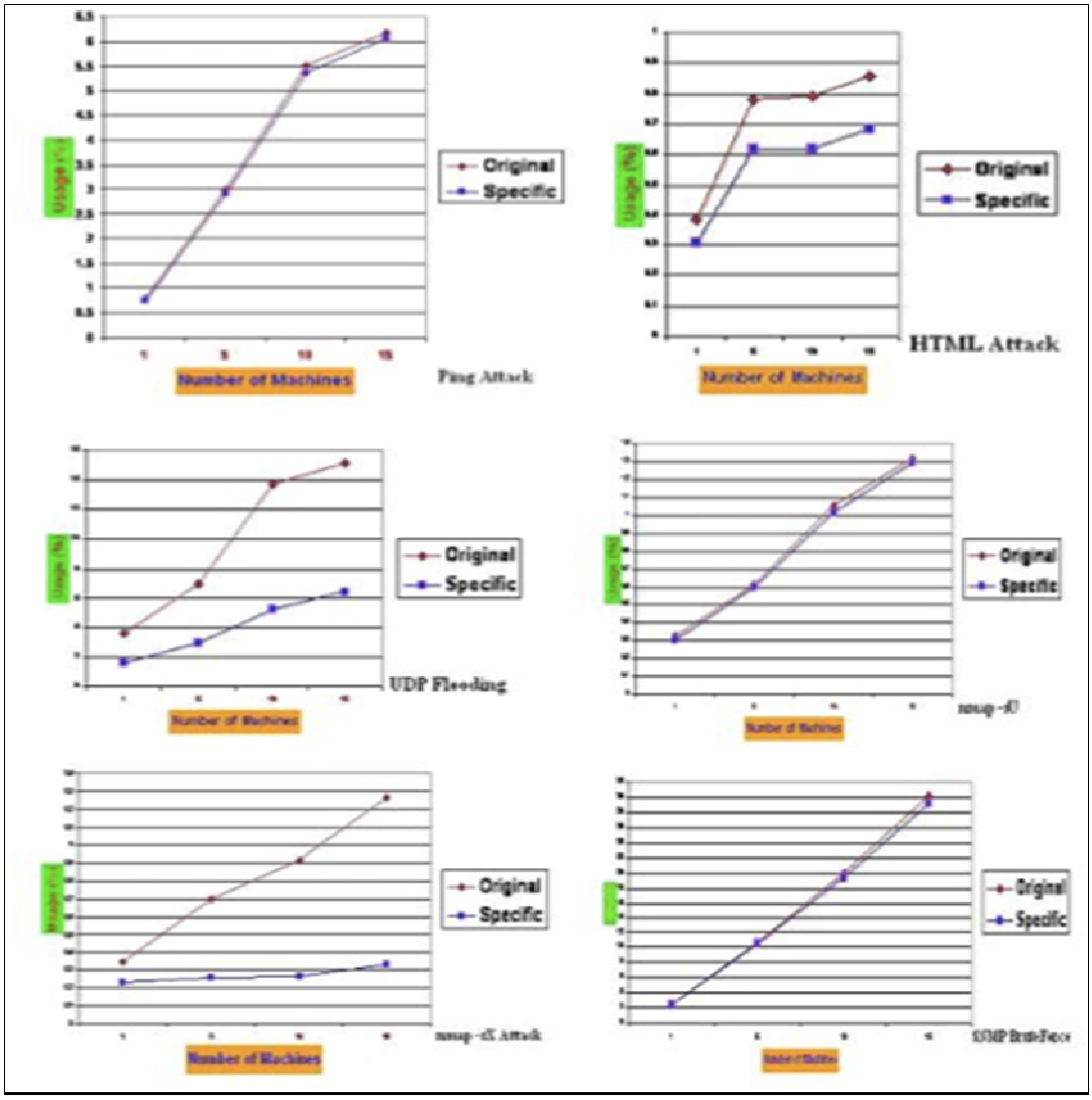


Figure 3.11 Effects of Snort Rules Set under Various Attacks

3.4 Performance Evaluation Study of Intrusion Detection System [1]

This paper tested and analyzed the performance of two well-known IDS systems, Snort and Suricata, at the high-speed network. They used three different platforms: ESXi (virtual machine), Linux 2.6 (Ubuntu10.10) and FreeBSD v.8.1 to compare the performance and give a suggestion of the operating system for each IDS at a specific traffic rate. Both systems used the default configuration and were tested under the same conditions as shown in Figure 3.12. Snort installed was version 2.9.0.4 and Suricata installed was version1.0.2.

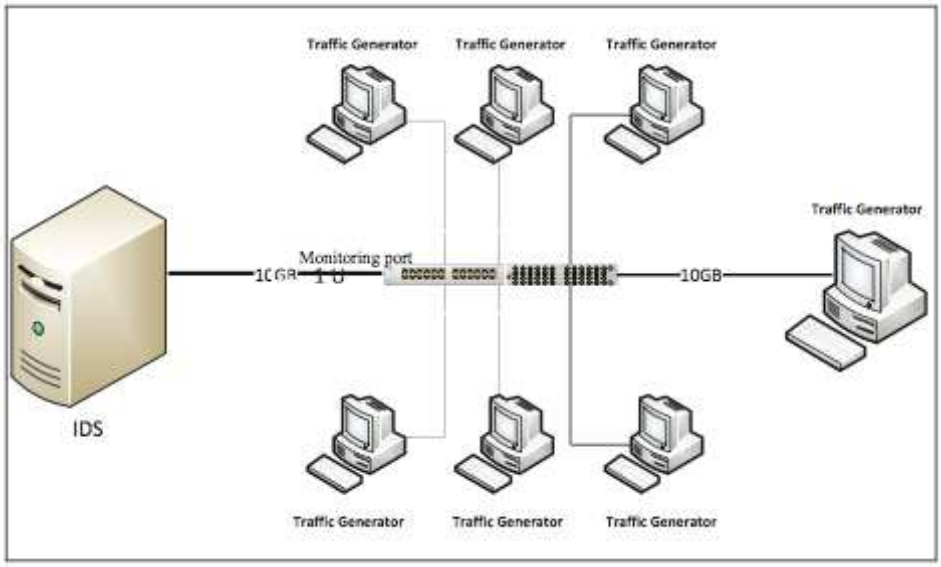


Figure 3.12 Testing Environment

The testing scenario was as follows:

- Snort and Suricata were installed on three different operating systems: Linux over ESXi, Linux, and FreeBSD.
- All scenario were tested using different packet sizes (1470, 1024, 512) for both TCP and UDP.
- The speed range was varied from 250Mbps, 500Mbps, 750Mbps, 1.0Gpbs, 1.5Gbps, and 2.0Gbps.
- In all the scenarios, Suricata and Snort were configured to load and run similar number of rules to monitor.

The results were displayed for TCP and UDP traffic for the performance of both Snort and Suricata. They were tested using different packet sizes and different speeds, while the percentage of packet drops were measured. The results are shown in Figure 3.13 and 3.14.

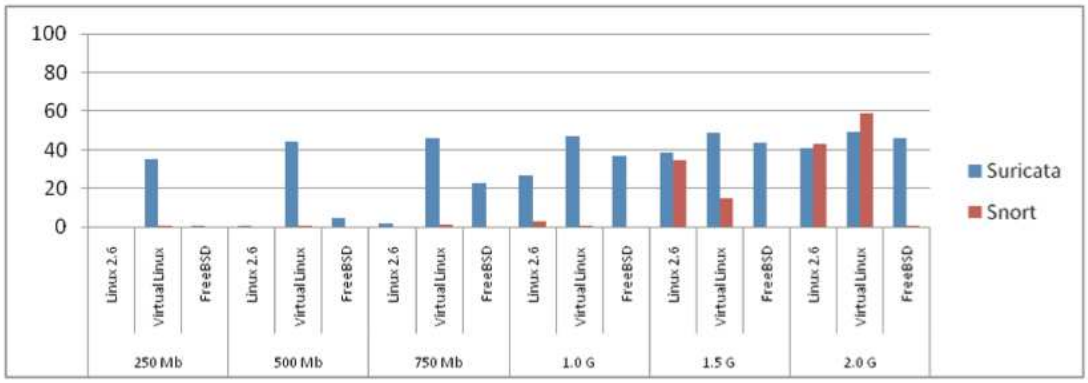


Figure 3.13 Snort and Suricata using Packet Size of 512 on TCP Traffic

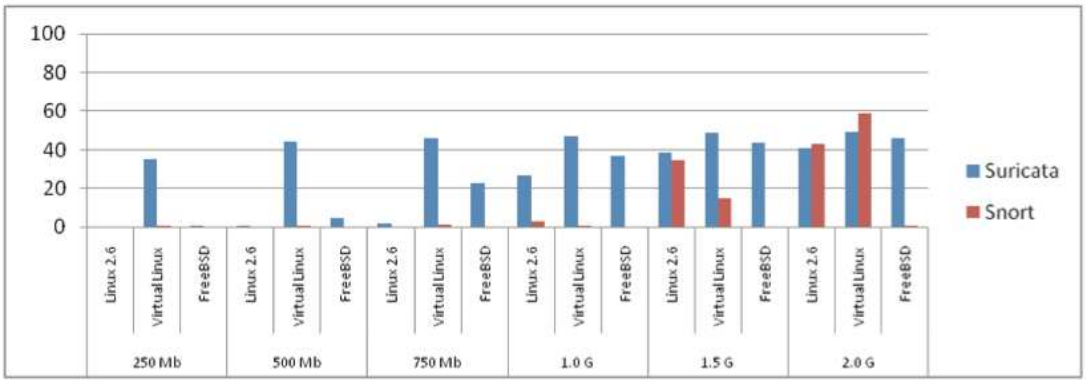


Figure 3.14 Snort and Suricata using Packet Size of 512 on UDP Traffic

Table 3.2 shows the CPU utilization and the packet drops for both Snort and Suricata on Linux and FreeBSD. Table 3.3depicted the percentage of alerts detected by both Snort and Suricata, while Table 3.4gives the suggestions of the operating system for use on each IDS. We can conclude that Suricata gave better performance on FreeBSD, especially when running on the high-speed network traffic.

Table 3.2 CPU Utilization and Packet Drops

Platform	Snort		Suricata	
	CPU Usage	Packet drops	CPU Usage	Packetdrops
Linux	27%	31.43%	68%	8.9
FreeBSD	21%	3.24%	24.5%	43.6

Table 3.3 Percentage of Alerts Detected

Speed	Snort	Suricata
1.0Gbps	100%	98%
1.5.Gbps	100%	91.8%
2.0Gbps	99.7%	66.8%

Table 3.4 Suggestions of Operating System for each IDS

IDSs Ideal operating systems (TCP traffic – Packed size 1024)

Speed	Suricata Ideal platform	Snort Ideal Platform
500	Linux 2.6	Linux 2.6 or FreeBSD
750	Linux 2.6	Linux 2.6 or FreeBSD
1.5	Linux 2.6	FreeBSD
2.0	Linux 2.6	FreeBSD

IDSs Ideal operating systems (UDP traffic – Packed size 1024)

Speed	Suricata Ideal platform	Snort Ideal Platform
500	Linux 2.6	Linux 2.6 or FreeBSD
750	Linux 2.6	Linux 2.6
1.5	Linux 2.6	FreeBSD
2.0	Linux 2.6	FreeBSD

3.5 Analysis and Evaluation of the Snort and Bro Network Intrusion Detection Systems [11]

This paper analyzed and evaluated the Snort and Bro over data cell phone networks. The data set for mobile traffic was given as follows.

- RTT-1M, 1 million packets of real traffic taken from a link
- RTT-5M, 1 million packets taken from the same link

The rule set used was the full rule set of VRT via the Snort2Bro software. The evaluation measures the number of alerts, and the analysis time. The experimental results shown in Figure 3.15 and 3.16 illustrated that Bro used the double time for what needed by Snort to do the analysis since the rules must be converted and they may not be perfect. Thus, Bro may work harder than Snort.

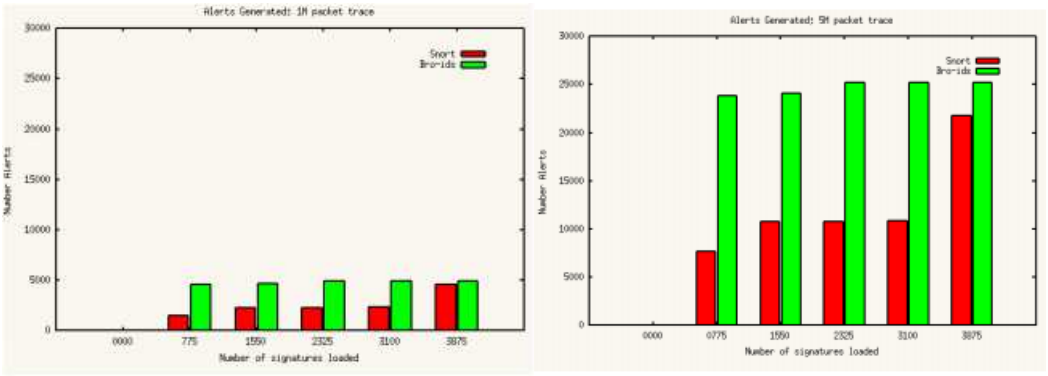


Figure 3.15 Alerts Generated for RTT-1M and RTT-5M

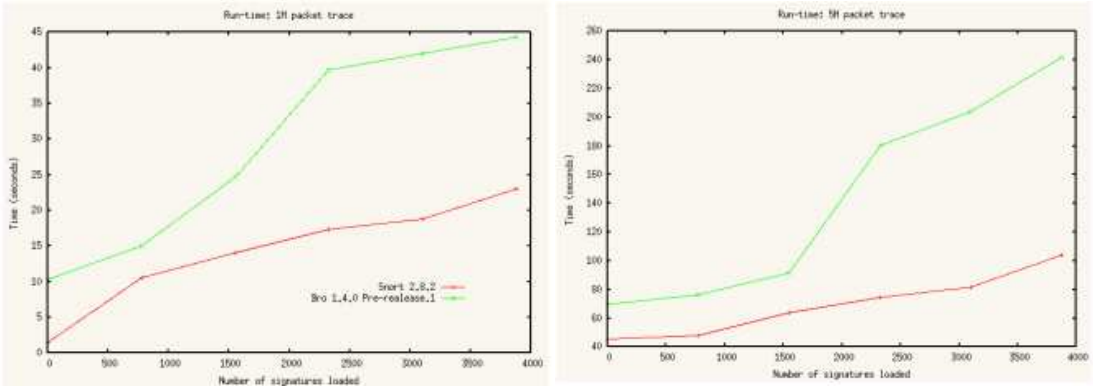


Figure 3.16 Run-time of traffic RTT-1M and RTT-5M

3.6 Summary of Reviewed Papers

Table 3.5 and Table 3.6 give the comparison of the five reviewed papers. We compare the versions of each IDS tool, the attack type tested, the number of rules, and the performance metrics used in those papers.

Table 3.5 Versions and Attack Types in the Five Papers

Paper	IDS tools			Attack						
	snort (v.)	bro	suricata	scan	HTML	dos/ddos	SNMP	icmp	dns	Other
1	✓(v.2.8.1)	X	X	X	X	X	X	X	X	✓
2	✓(v.2.8.0)	X	X	✓	X	X	X	X	X	X
3	✓(v.2.4.x)	X	X	✓	✓	✓	✓	✓	X	X
4	✓(v.2.9.0)	X	✓(v.1.0.2)	X	X	X	X	X	X	✓
5	✓(v.2.4.x)	✓(v.1.1.x)	X	X	X	X	X	X	X	✓
our proposed work	✓(v.2.9.1)	✓(v.1.5.3)	✓(v.1.1.1)	✓	✓	✓	✓	✓	✓	X

Table 3.6 Number of Rules and Metrics Measured in the Five Papers

Paper	number of rules	metric			
		packets loss	throughput	resource utilization	Alert
1	1	✓(loss probability)	✓(number of packets analyed)	X	X
2	All	✓(Numbers of packets)	X	CPU	X
3	All/ Some	X	X	CPU/Disk	✓(Disk Usage)
4	All	✓(loss percentage)	X	X	✓(undefined)
5	All	X	✓(time of packets analyed)	X	✓(Numbers of Alerts)
our proposed work	All/ Some	✓(Numbers of packets)	✓(number of packets analyed)	CPU	✓(Numbers of Alerts)

CHAPTER IV
PROPOSED WORK

This chapter presents the proposed model of our work which consists of three parts. The first part is the system architecture used to describe the evaluation model. The second part is the IDS rules comparison, and the third part is the network traffic generator tools.

4.1 System Architecture

Our proposed model as shown in Figure 4.1 has three parts: the generated input traffic, IDS software and its related rules used, and the alert outputs. The first part is the input network traffic generated by network traffic generator. They are sent to each IDS via the tcpdump. The second part is the IDS tool having a set of active rules to detect an intrusion. The three IDS tools chosen are Snort Bro and Suricata, which all are installed with the official configuration. The last part is the alert outputs generated by each IDS. They will illustrate the effects of the responsible rule sets to a specific network traffic for each IDS as the number of alerts will be reported.

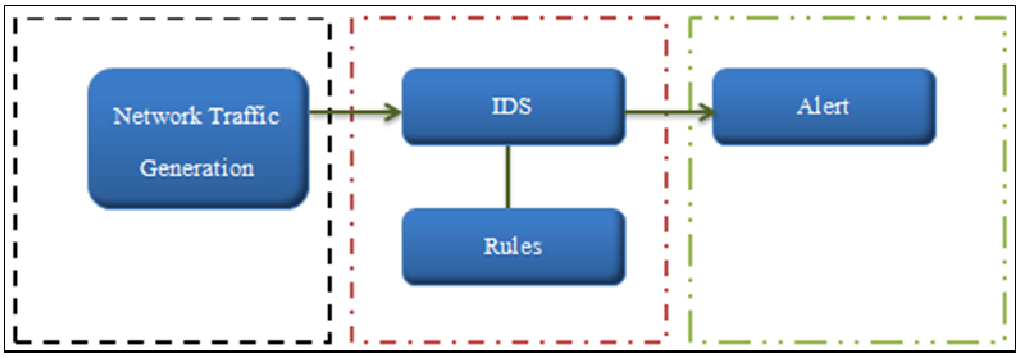


Figure 4.1 Overview of the Proposed Model

4.2 IDS Rules

In this research, we use the rule sets from “Emerging Threats”, an open source community project, which provides the standard rule sets for Snort and Suricata as well as gives the official rule sets of Bro. Table 4.1 shows the comparison of all rules whether they are present or not present in each IDS. Obviously, Snort and Suricata have every rule in common, whereas Bro lacks lots of rules to handle many attacks. We have analyzed further and found the shared rules among all three IDS as displayed in Table 4.2.

Table 4.1 Rules Comparison among Snort, Suricata and Bro

Rule Types	Snort	Suricata	Bro	Rule Types	Snort	Suricata	Bro
bad-traffic	✓	✓	×	attack-responses	✓	✓	×
exploit	✓	✓	×	oracle	✓	✓	×
scan	✓	✓	✓	mysql	✓	✓	×
finger	✓	✓	✓	snmp	✓	✓	✓
ftp	✓	✓	✓	smtp	✓	✓	×
telnet	✓	✓	✓	imap	✓	✓	✓
rpc	✓	✓	×	pop2	✓	✓	×
rservices	✓	✓	×	pop3	✓	✓	✓
dos	✓	✓	✓	nntp	✓	✓	×
ddos	✓	✓	✓	web-attacks	✓	✓	×
dns	✓	✓	✓	backdoor	✓	✓	×
tftp	✓	✓	×	shellcode	✓	✓	×
web-cgi	✓	✓	×	policy	✓	✓	×
web-coldfusion	✓	✓	×	porn	✓	✓	×
web-iis	✓	✓	×	info	✓	✓	×
web-frontpage	✓	✓	×	icmp-info	✓	✓	×
web-misc	✓	✓	×	virus	✓	✓	×
web-client	✓	✓	×	chat	✓	✓	×
web-php	✓	✓	×	multimedia	✓	✓	×
sql	✓	✓	×	p2p	✓	✓	×
x11	✓	✓	×	spyware-put	✓	✓	×
ssh	✓	✓	✓	specific-threats	✓	✓	×
icmp	✓	✓	✓	experimental	✓	✓	×
netbios	✓	✓	×	content-replace	✓	✓	×
misc	✓	✓	×	voip	✓	✓	×

Table 4.2 Shared Rules of the Three IDS

scan	ssh
finger	icmp
ftp	snmp
telnet	dns
dos	imap
ddos	pop3

From the shared rule sets given in Table 4.2, we select eight types of attacks and count the number of rules for those attacks as displayed in Table 4.3

Table 4.3 Number of Rules

No.	Rule Type	Number of Rules
1	DNS	35
2	DoS/DDoS	89
3	FTP	93
4	ICMP	105
5	POP3/IMAP	114
6	SCAN	20
7	SNMP	19
8	Telnet/SSH	21

For better understanding of rules, we present a set of examples for the shared rule set as they can apply to Snort and Suricata. However, the Snort2Bro tool will be used to convert them for use in Bro. The sample rules for five common attacks are explained below.

1. SCAN Attack

The rule as shown in Figure 4.2 generates the event when a scan for the version of an ssh daemon is detected. The event indicates that an attempt has been made to scan a host. Particularly, an attempt has been made to scan for the version of the ssh daemon on the target host.

```
alerttcp $EXTERNAL_NET any -> $HOME_NET 22 (msg:"SCAN SSH Version
map attempt"; flow:to_server,established; content:"Version_Mapper";
fast_pattern:only; classtype:network-scan; sid:1638; rev:6;)
```

Figure 4.2 Example of Snort Rule in Scan Attack Type

2. DNS Attack

The rule as shown in Figure 4.3generates the event when an attempt is made to query “version bind” on the DNS server. An attacker can execute this query to find a DNS server running specific versions of BIND.

```
alert udp $EXTERNAL_NET any -> $HOME_NET 53 (msg:"DNS named version
attempt"; flow:to_server; content:"|07|version"; offset:12; nocase;
content:"|04|bind|00|"; offset:12; nocase; metadata:servicedns;
reference:arachnids,278; reference:nessus,10028; classtype:attempted-recon;
sid:1616; rev:11;)
```

Figure 4.3 Example of Snort Rule in DNS Attack Type

3. DOS Attack

The rule as shown in Figure 4.4generates the event when a remote attacker transmits a malformed request for a page on a web server port which can indicate a Denial of Service (DoS) attack on a RealServer. An attacker sends an HTTP request for “/viewsource/template.html?” on a RealServer audio server. When the RealServer crashes, the audio transmission will be stopped.

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 8080 (msg:"DOS Real Server
template.html"; flow:to_server,established; content:"/viewsource/template.html?";
fast_pattern:only; reference:bugtraq,1288; reference:cve,2000-0474;
classtype:attempted-dos; sid:278; rev:10;)
```

Figure 4.4 Example of Snort Rule in DoS Attack Type

4. DDOS Attack

The rule as shown in Figure 4.5generates the event when a ping packet for the Trinoo also known as trin00” DDoS suite is detected. As part of a large scale attack against a machine or a network, an attacker will compromise a large number of machines which will form the army that the trin00 master daemon will command. The master daemon typically instructs the clients to send the mass quantities of packets to a

set of victim hosts. If the traffic is sufficient, the victim machines will become resource deprived and thus endure a DoS condition.

```
alert udp $EXTERNAL_NET any -> $HOME_NET 31335 (msg:"DDOS Trin00
Daemon to Master *HELLO* message detected"; flow:to_server;
content:"*HELLO*"; reference:arachnids,185; reference:cve,2000-0138;
reference:url,www.sans.org/newlook/resources/IDFAQ/trinoo.htm;
classtype:attempted-dos; sid:232; rev:10;)
```

Figure 4.5 Example of Snort Rule in DDoSAttack Type

5. ICMP Attack

The rule as shown in Figure 4.6 generates the event when an ICMP echo request is made from a host running the L3 "Retriever 1.5" security scanner. An attacker may attempt to determine live hosts in a network prior to launching an attack.

```
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP L3retriever
Ping"; icode:0; itype:8; content: "ABCDEFGHIJKLMNOPQRSTUVW
ABCDEFGHI"; depth:32; reference:arachnids, 311;classtype:attempted-recon;
```

Figure 4.6Example of Snort Rule in ICMP Attack Type

4.3 Network Traffic Generator Tools

In our work, the network traffic is generated using four traffic generator tools: Ostinato, Network Mapper (NMAP), High Orbit Ion Cannon (HOIC), and Low Orbit Ion Cannon (LOIC). Each tool is briefly described below.

1. Ostinato

Ostinato [15] is an open-source, cross-platform packet/traffic generator and analyzer with friendly graphic interfaces as shown in Figure 4.7. The followings are the main features of Ostinato.

- Support the most common standard protocols: SNAP, TCP, UDP, and ICMP

- Modify any field of any protocols
- Create and configure multiple streams
- Configure stream rates, bursts, and the number of packets
- A single client can control and configure multiple ports on multiple computers generating traffic
- Controlling of a port to prevent the OS from sending stray packets provides a controlled testing environment
- Statistics window shows the real time port receive/transmit statistics and rates
- Capture packets and view them via the software like Wireshark

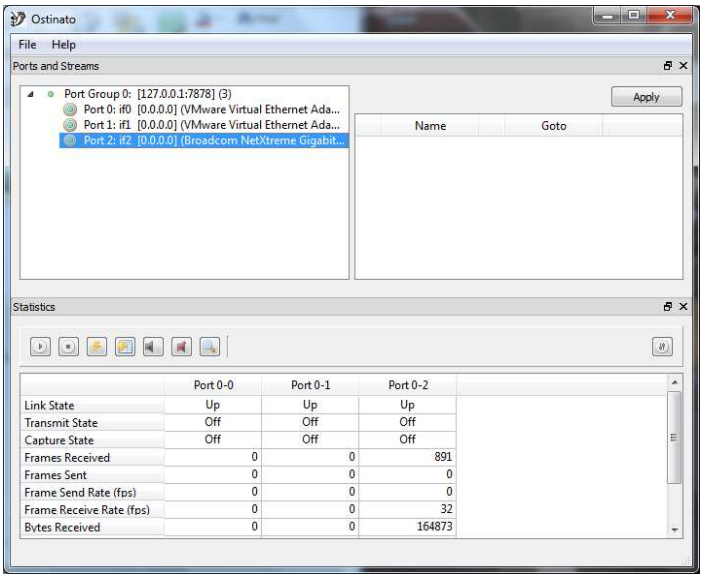


Figure 4.7 Ostinato Graphic User Interface

Figure 4.8 gives an example of the generated DDoS attack using random source IP addresses, MAC addresses, the payload size, transmitted over Ethernet, IPv4, TCP/IP protocol, the port 80, and the speed 100 pps.

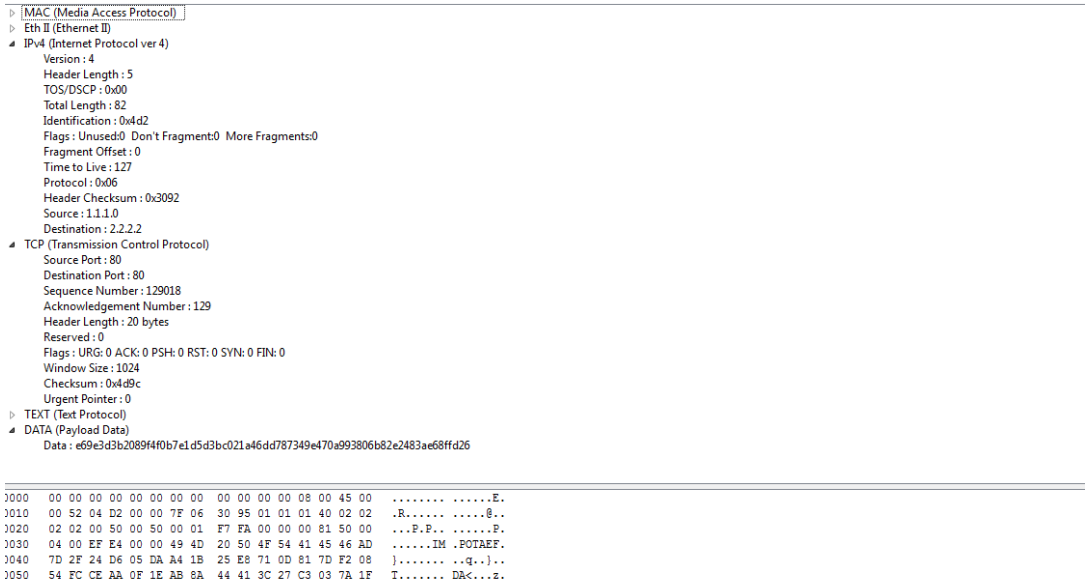


Figure 4.8 Example of Packet Generated by Ostinato

2. Network Mapper (NMAP)

NMAP [13] is an open source tool for network discovery and security auditing. Many systems and network administrators find it useful for other tasks such as network inventory, managing service upgrade schedules, host monitoring and service uptime. In addition, NMAP uses raw IP packets in a novel way to determine what hosts are available on the network, what services those hosts are offering, what operating systems they are running, what type of packet filters/firewalls are in use, and dozens of other characteristics. It was designed to rapidly scan large networks, but works fine against single hosts. With these penetrated properties to networks and hosts, NMAP has also been used as attacking tools in many occasions. The graphic user interfaces of NMAP are shown in Figure 4.9.

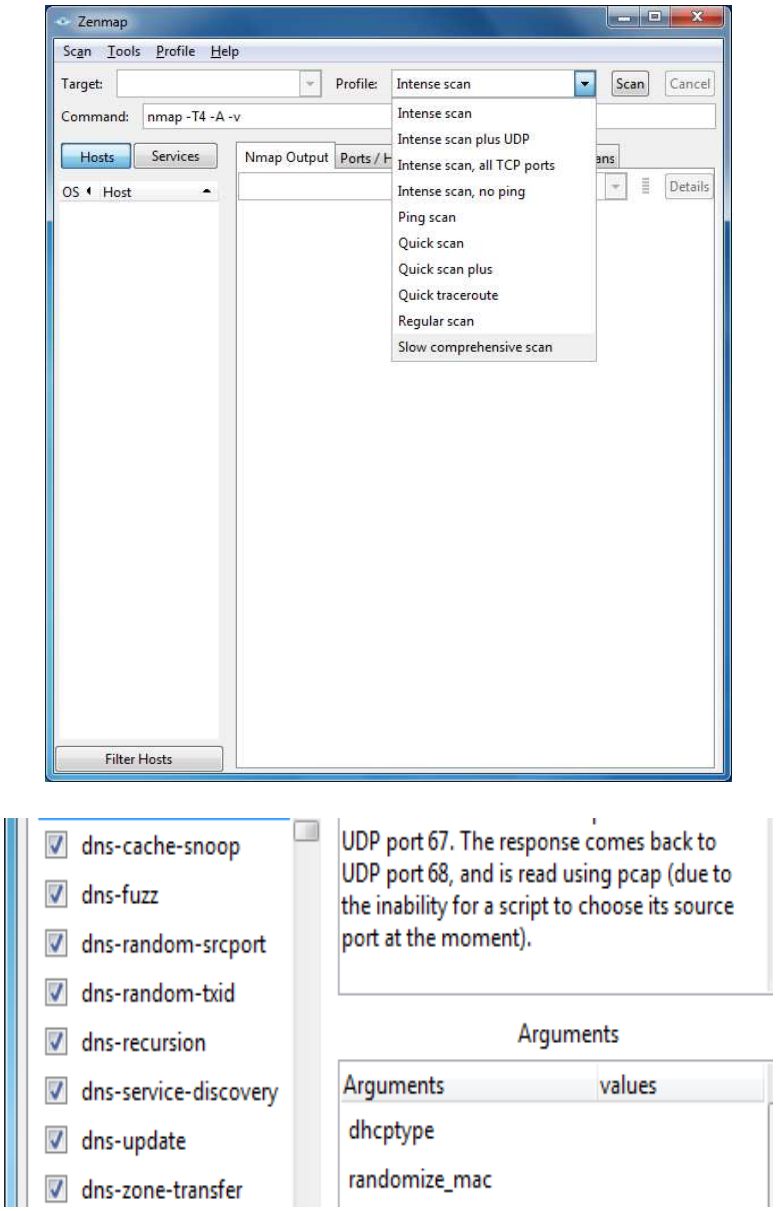


Figure 4.9 NMAP Graphic User Interface

3. Low Orbit Ion Cannon (LOIC)

LOIC [9] attacks a target site by flooding a server with TCP packets and UDP packets. It also has the function to perform the DoS attacks, but uses the TCP/UDP packets to flood the target. But, it is easy to bypass the attacks from LOIC by writing good firewall rules. For a positive manner, LOIC can be used as a network stress checking tool. The graphic user interface of LOIC is shown in Figure 4.10.

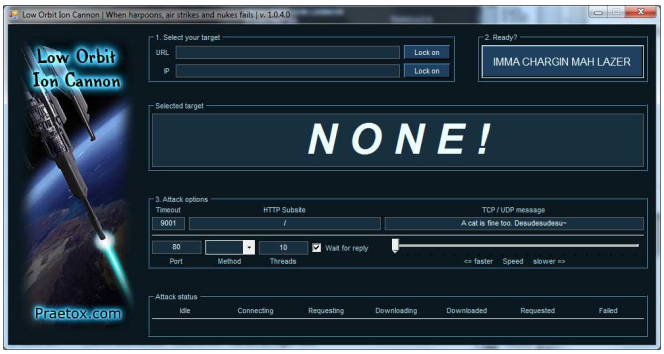


Figure 4.10 Low Orbit Ion Cannon GUI

4. High Orbit Ion Cannon (HOIC)

HOIC [5] generates a high-speed multi-threaded HTTP flood attack. The followings are the main features of HOIC.

- High-speed multi-threaded HTTP Flooding
- Simultaneously flood up to multiple websites at once
- Scripted “Boosters” to handle DDoS counter measures and increase DoS output.
- Generating multiple HTTP headers to create the genuine traffic flow scenario

The graphic user interface of HOIC is shown in Figure 4.11.

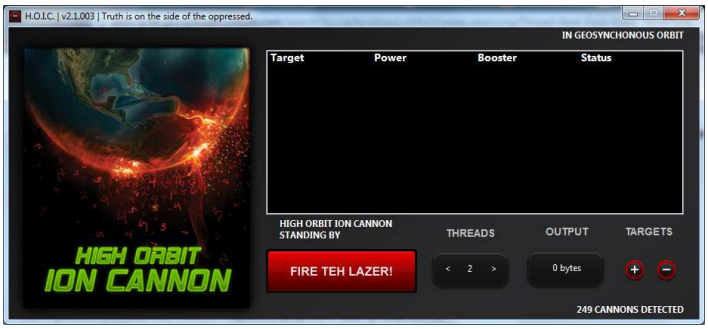


Figure 4.11 High Orbit Ion Cannon GUI

CHAPTER V
EXPERIMENTS AND RESULTS

5.1 Experimental Setup

Figure 5.1 below shows the testing environment of our work to measure the accuracy and the performance of the three IDS. The system configuration consists of (1) the network traffic generator for both background and malicious traffic, (2) the network switch, (3) the target server, and (4) the IDS machines for Snort, Bro and Suricata. The packet or traffic generator is the Ostinato tool which generates packets and sends to the IDS. The generator generates a specific traffic rate sent to the receiver and the performance is measured at the receiver. The traffic generator also generates packets of an attack type and sends to the IDS. The IDS will measure the accuracy.

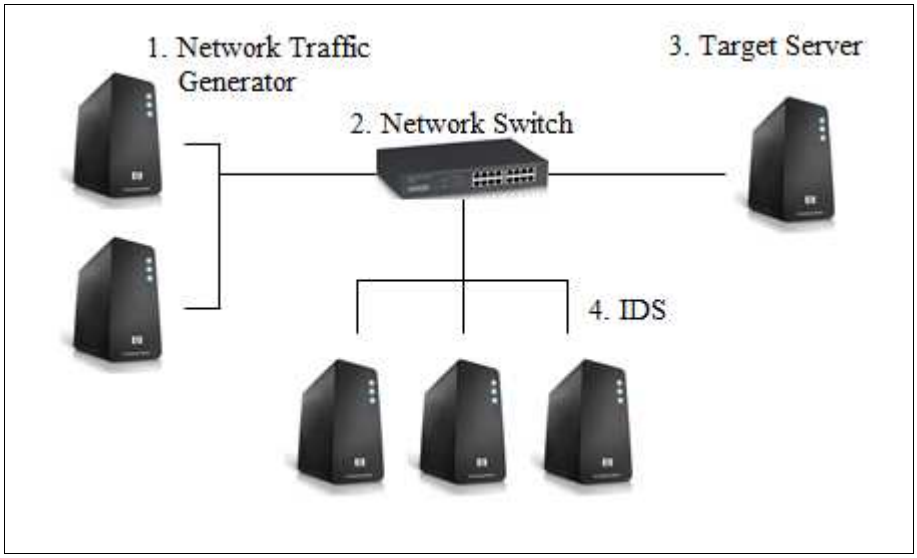


Figure 5.1 System Configuration

1.2 Hardware and Software

Table 5.1 presents the hardware and the software configuration of the machines used in our experiments.

Table 5.1 Hardware and Software Configuration

Traffic Generator		**Target Server**	**3 IDS Machines**
Hardware	**Hardware**	**Hardware**	**Hardware**
Dell Optiplex GX520	Acer veritron S661	Dell Optiplex GX520	Acer veritron S661
Intel® 3.4GHz Duo processor	Intel® 2.3GHz	Intel® 3.4GHz Duo processor	Intel® 2.3GHz
RAM 2GB	RAM 1GB	RAM 2GB	RAM 1GB
HD 80GB	HD 80GB	HD 80GB	HD 80GB
Software	**Software**	**Software**	**Software**
Windows 7	Cent OS 5	Cent OS 5	Cent OS 5
Ostinato 0.3	Tcpreplay 3.4.3	Bind 9.6	Server1,Snort 2.9.1
LOIC 1.0.4		Openssh 5.1	Server2,Bro-ids 1.5
HOIC 2.1		Net-snmp 5.4.1	Server3,Suricata 1.1
NMAP (Zenmap 5.51)		Apache2 2.2.9	
Wireshark 1.4.2			

5.3 Performance Metrics

To evaluation the IDS, the following three metrics are used.

1. *Number of packets loss*

The total number of packets dropped by IDS is equal to the total number of packets sent by the packet generator subtracted by the total number of packets received by IDS.

2. *Number of alerts generated by each IDS*

The information from an alert log is gathered to compute the number of alerts and the type of alerts.

3. *CPU and Memory usage of the IDS process*

The *top* command is used to gather the information of the resource utilization from the IDS.

5.4 Experiment 1 and Results

The first experiment evaluates the performance of each IDS under a variety of traffic rates and protocols. The experiment measures the numbers of packets loss, CPU utilization, memory usage and the number of alerts. The shared rule set is used, and one million packets of normal and malicious traffic are analyzed. Table 5.2 gives the values of parameters set for Experiment 1.

Table 5.2 Experiment 1Parameters

Normal traffic	
Traffic rate Protocol	50,100,200,300,400,500,600,700,800,900,1000,2000 pps TCP,UDP
Malicious traffic	
Traffic rate All of Attack types	50,100,200,300,400,500,600,700,800,900,1000,2000 pps DNS attack, DoS/DDoS attack, FTP attack, ICMP attack, POP3/IMAP attack, SCAN attack, SNMP Attack and Telnet attack.
Combined traffic	
Background traffic rate Malicious traffic rate Three attack types	Fixed traffic rate at 300 pps for TCP traffic 100,200,300,400,500,600,700 pps DoS/DDoS, SCAN and SNMP
Rule set	Shared rule set

5.4.1 Results of Experiment 1 for Normal Traffic

Figure 5.2shows the results when the normal traffic TCP is analyzed. In addition, Figure 5.2 (a) presents the average memory usage where the differences among the three IDS are high. Suricata and Snort has higher average memory usage of 17.3% and 20.3 %, respectively, whereas Bro has the lowest memory usage of 1.80 %.Figure 5.2 (b) presents the CPU usage where all IDS show the same trend of increasing for higher packet rates. However, Snort and Suricata gave the slightly

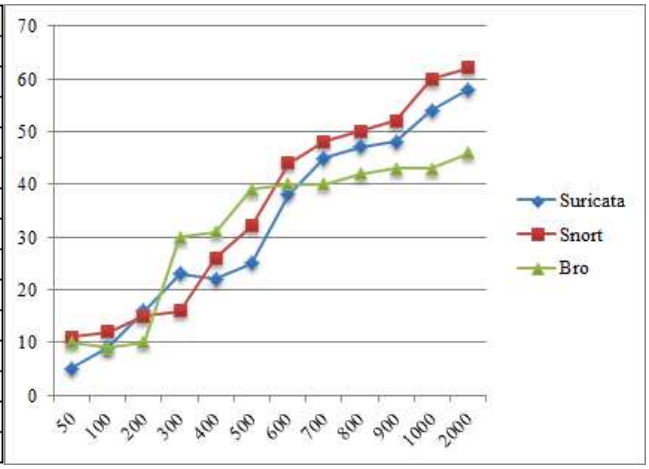
higher increasing trend in the CPU usage than Bro. Figure 5.2(c) presents the numbers of packet loss where, again, Bro gives the lowest packet loss.

(a) Memory Usage

IDS	%
Suricata	17.3
Snort	20.3
Bro	1.8

(b) CPU Usage

	Suricata	Snort	Bro
Traffic Rate	CPU	CPU	CPU
PPS	%	%	%
50	5	11	10
100	9	12	9
200	16	15	10
300	23	16	30
400	22	26	31
500	25	32	39
600	38	44	40
700	45	48	40
800	47	50	42
900	48	52	43
1000	54	60	43
2000	58	62	46



(c) Packet Loss

	Suricata	Snort	Bro
Traffic Rate	Packets Loss		
PPS	%	%	%
50	0.0	0.0	0.0
100	0.0	0.0	0.0
200	0.0	0.0	0.0
300	0.0	0.7	9.0
400	0.0	0.9	9.6
500	0.4	10.1	12.1
600	0.4	10.1	12.5
700	0.7	17.1	18.3
800	8.5	15.6	22.7
900	9.4	30.5	22.9
1000	33.6	35.4	24.2
2000	43.3	45.2	31.4

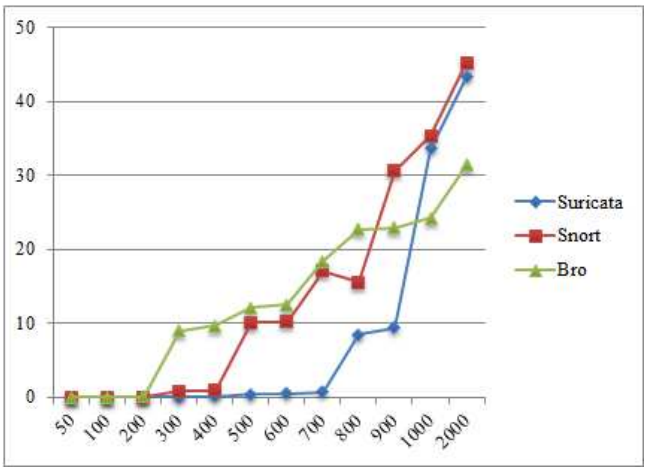


Figure 5.2 Results of Normal TCP Traffic

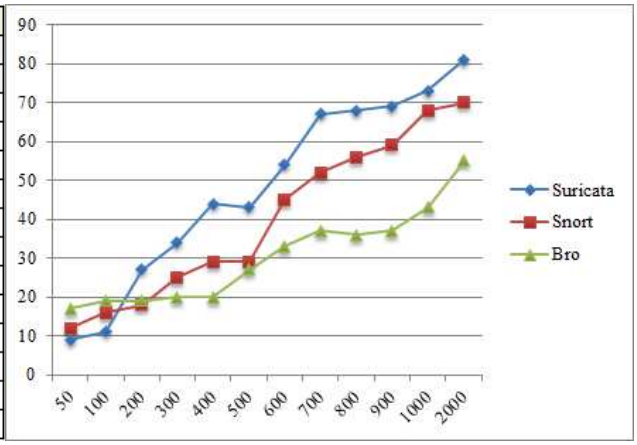
Figure 5.3 shows the results when the normal traffic UDP is analyzed. Figure 5.3(a) presents the average memory usage where Bro has the lowest memory usage. Figure 5.3 (b) shows the similar trend of CPU usage as the TCP traffic, and Bro gives the lowest CPU usage. Figure 5.3(c) presents the number of packet loss where Suricata gives the consistent but high packet loss and Bro gives lower packet loss except the sharp increasing trend at the high traffic rate.

(a) Memory Usage

IDS	%
Suricata	17.3
Snort	20.4
Bro	1.8

(b) CPU Usage

	Suricata	Snort	Bro
Traffic Rate	CPU	CPU	CPU
PPS	%	%	%
50	9	12	17
100	11	16	19
200	27	18	19
300	34	25	20
400	44	29	20
500	43	29	27
600	54	45	33
700	67	52	37
800	68	56	36
900	69	59	37
1000	73	68	43
2000	81	70	55



(c) Packet Loss

	Suricata	Snort	Bro
Traffic Rate	Packets Loss		
PPS	%	%	%
50	1.0	0.0	0.0
100	15.8	0.0	0.1
200	17.7	0.0	0.1
300	17.8	0.0	5.1
400	18.0	5.4	8.0
500	18.4	8.3	10.5
600	18.5	15.2	13.6
700	18.7	19.7	15.9
800	22.7	25.4	15.4
900	21.8	27.8	17.6
1000	20.1	28.4	21.5
2000	24.1	30.9	33.8

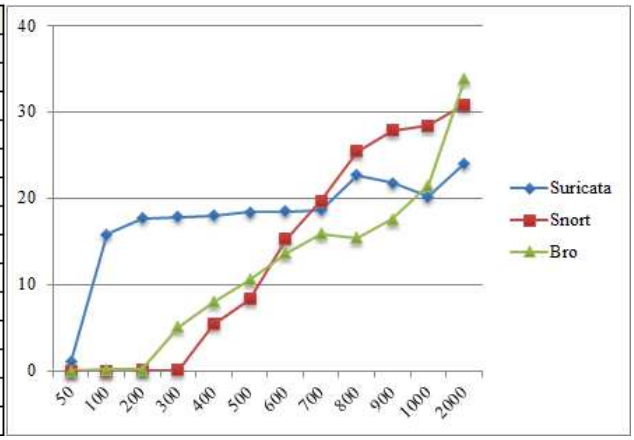


Figure 5.3 Results of Normal UDP Traffic

5.4.2 Results of Experiment 1for Malicious Traffic

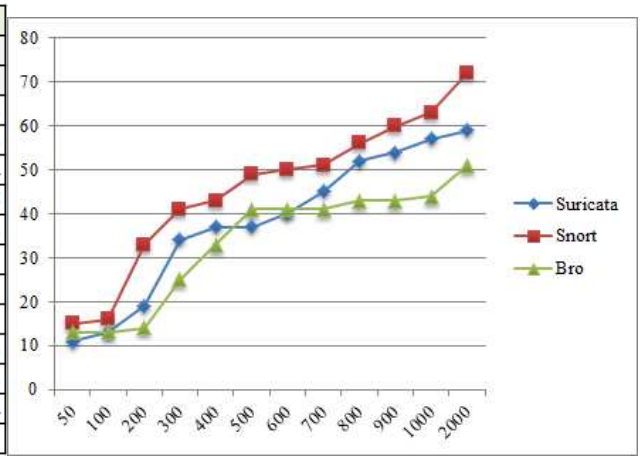
Figure 5.4shows the results when the eight groups of malicious traffic have been analyzed. The eight attacks are DNS attack, DoS/DDoS attack, FTP attack, ICMP attack, POP3/IMAP attack, SCAN attack, SNMP attack and Telnet attack.

(a) Memory Usage

IDS	%
Suricata	17.3
Snort	20.2
Bro	18.7

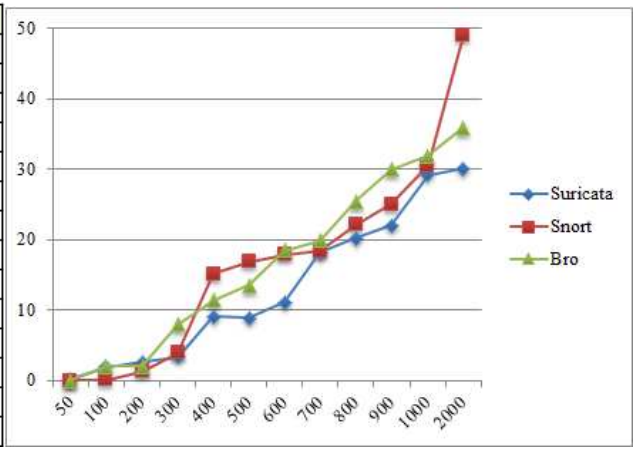
(b) CPU Usage

	Suricata	Snort	Bro
Traffic Rate	CPU	CPU	CPU
PPS	%	%	%
50	11	15	13
100	13	16	13
200	19	33	14
300	34	41	25
400	37	43	33
500	37	49	41
600	40	50	41
700	45	51	41
800	52	56	43
900	54	60	43
1000	57	63	44
2000	59	72	51



(c) Packet Loss

	Suricata	Snort	Bro
Traffic Rate	Packets Loss		
PPS	%	%	%
50	0.1	0.0	0.0
100	1.9	0.1	2.0
200	2.6	1.3	2.0
300	3.3	3.9	8.0
400	9.0	15.1	11.4
500	8.9	16.8	13.5
600	11.0	17.8	18.5
700	18.2	18.4	19.8
800	20.2	22.1	25.4
900	22.0	25.0	29.9
1000	29.2	30.5	31.9
2000	30.1	49.0	35.9



(d) Alert Numbers

	Suricata	Snort	Bro
Traffic Rate	Alert	Alert	Alert
PPS	#	#	#
50	1298	890	1005
100	1298	890	981
200	1291	875	981
300	1280	874	981
400	1280	831	937
500	1280	827	936
600	1267	825	930
700	1253	825	930
800	1201	801	930
900	1190	801	927
1000	1092	793	927
2000	927	750	854

(e) Average Alert Numbers

IDS	#
Suricata	1221
Snort	832
Bro	943

Figure 5.4 Results of DNS Attack Traffic

Figure 5.4 (a) shows the similar average memory usage among all three IDS for DNS attack. Figure 5.4 (b) shows that the CPU usage has the increasing trends but Bro gives the lowest CPU usage. Figure 5.4 (c) presents the similar increasing trend of the number of packet loss except that Snort has the high sharp increase at the end. Figure 5.4 (d) and (e) show that the number of alerts in Suricata is the highest while Snort gives the lowest number of alerts.

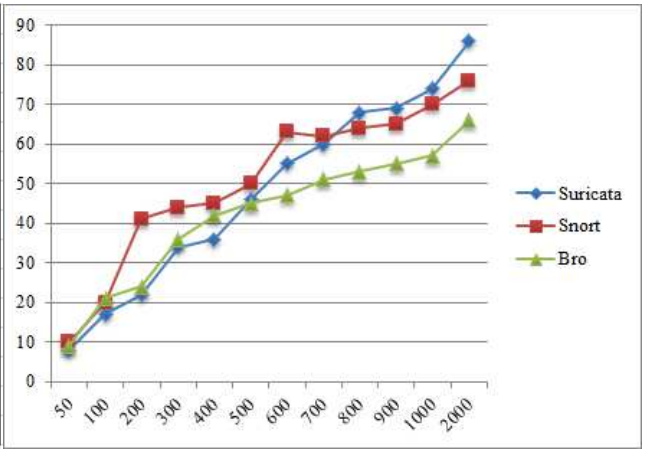
Figure 5.5 (a) shows the similar average memory usage among all three IDS for DoS/DDoS attack where Bro has the lowest memory usage. Figure 5.5 (b) shows that the CPU usage has the increasing trends where Bro gives the lowest CPU usage. Figure 5.5 (c) also presents the similar increasing trend of the number of packet loss where Suricata has the highest packet loss. Figure 5.5 (d) and (e) show that the number of alerts in Suricata is the highest while Snort gives the lowest number of alerts.

(a) Memory Usage

IDS	%
Suricata	17.2
Snort	20.3
Bro	17.9

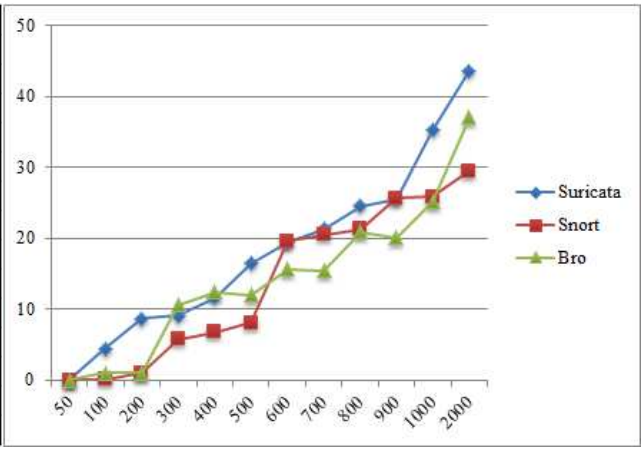
(b) CPU Usage

	Suricata	Snort	Bro
Traffic Rate	CPU	CPU	CPU
PPS	%	%	%
50	8	10	9
100	17	20	21
200	22	41	24
300	34	44	36
400	36	45	42
500	46	50	45
600	55	63	47
700	60	62	51
800	68	64	53
900	69	65	55
1000	74	70	57
2000	86	76	66



(c) Packet Loss

	Suricata	Snort	Bro
Traffic Rate	Packets Loss		
PPS	%	%	%
50	0.0	0.0	0.0
100	4.4	0.0	1.0
200	8.7	1.0	1.0
300	9.1	5.7	10.6
400	11.6	6.7	12.4
500	16.4	8.0	12.0
600	19.4	19.5	15.7
700	21.3	20.5	15.4
800	24.5	21.3	20.9
900	25.5	25.7	20.0
1000	35.2	25.8	25.1
2000	43.6	29.5	37.0



(d) Alert Numbers

	Suricata	Snort	Bro
Traffic Rate	Alert	Alert	Alert
PPS	#	#	#
50	1318	901	1045
100	1318	901	1045
200	1302	900	1004
300	1289	897	992
400	1289	890	992
500	1290	890	992
600	1289	890	990
700	1153	884	990
800	1106	876	950
900	998	876	947
1000	998	853	901
2000	915	810	856

(e) Average Alert Numbers

IDS	#
Suricata	1189
Snort	881
Bro	975

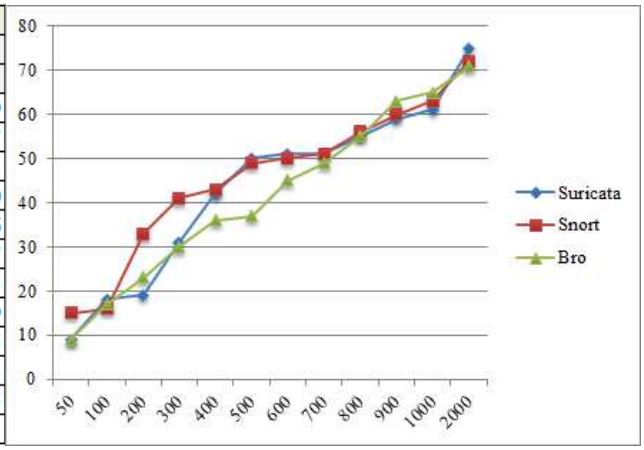
Figure 5.5 Results of DoS/DDoS Attack Traffic

(a) Memory Usage

IDS	%
Suricata	17.3
Snort	20.3
Bro	18.2

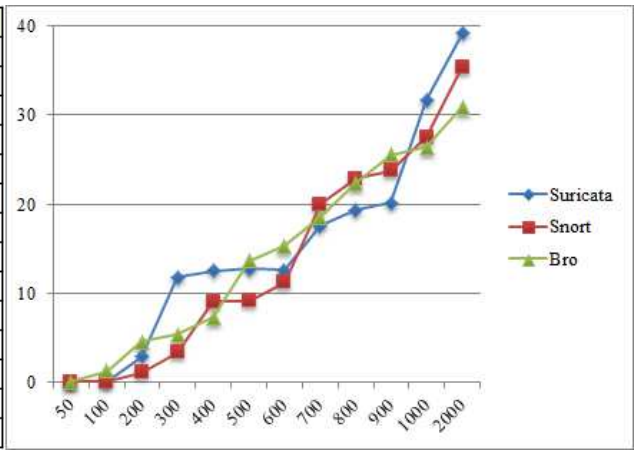
(b) CPU Usage

	Suricata	Snort	Bro
Traffic Rate	CPU	CPU	CPU
PPS	%	%	%
50	9	12	9
100	18	23	17
200	19	29	23
300	31	31	30
400	42	45	36
500	50	48	37
600	51	53	45
700	51	53	49
800	55	61	55
900	59	70	63
1000	61	71	65
2000	75	85	71



(c) Packet Loss

	Suricata	Snort	Bro
Traffic Rate	Packets Loss	Packets Loss	Packets Loss
PPS	%	%	%
50	0.0	0.0	0.0
100	0.0	0.0	1.2
200	2.8	1.1	4.5
300	11.7	3.3	5.3
400	12.5	9.0	7.3
500	12.7	9.1	13.6
600	12.6	11.2	15.3
700	17.5	20.0	18.6
800	19.3	22.9	22.4
900	20.2	23.7	25.5
1000	31.7	27.5	26.4
2000	39.2	35.4	30.9



(d) Alert Numbers

	Suricata	Snort	Bro
Traffic Rate	Alert	Alert	Alert
PPS	#	#	#
50	974	957	1023
100	974	957	1021
200	960	943	1013
300	942	920	1013
400	937	912	947
500	931	912	943
600	920	912	932
700	912	893	927
800	911	892	921
900	910	891	920
1000	910	890	920
2000	905	872	913

(e) Average Alert Numbers

IDS	#
Suricata	932
Snort	913
Bro	958

Figure 5.6 Results of FTP Attack Traffic

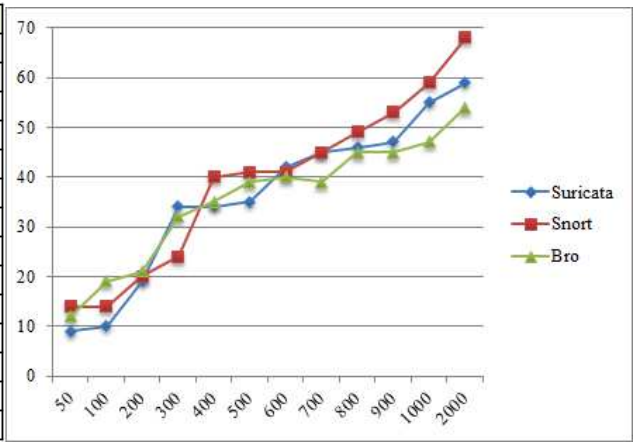
Figure 5.6 (a) shows the similar average memory usage among all three IDS for FTP attack where Suricata has the lowest memory usage. Similarly, Figure 5.6 (b) shows that the CPU usage has the increasing trends where Snort gives the highest trend. Figure 5.6 (c) presents the similar but mixed increasing trend of the number of packet loss. Figure 5.6 (d) and (e) also show that the similar number of alerts for all three IDS.

(a) Memory Usage

IDS	%
Suricata	17.3
Snort	20.3
Bro	18.5

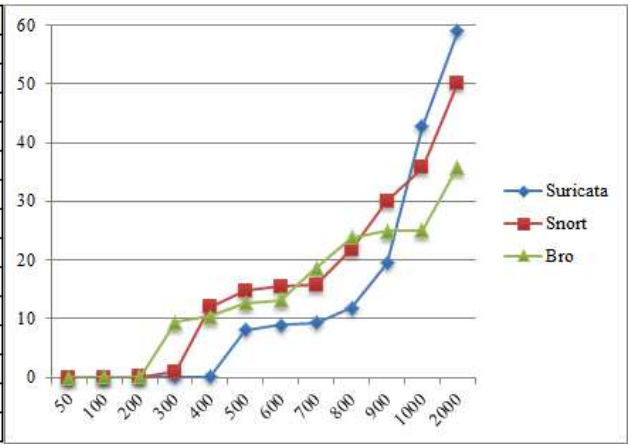
(b) CPU Usage

	Suricata	Snort	Bro
Traffic Rate	CPU	CPU	CPU
PPS	%	%	%
50	9	14	12
100	10	14	19
200	19	20	21
300	34	24	32
400	34	40	35
500	35	41	39
600	42	41	40
700	45	45	39
800	46	49	45
900	47	53	45
1000	55	59	47
2000	59	68	54



(c) Packet Loss

	Suricata	Snort	Bro
Traffic Rate	Packets Loss		
PPS	%	%	%
50	0.0	0.0	0.0
100	0.0	0.0	0.0
200	0.0	0.1	0.1
300	0.1	1.0	9.4
400	0.1	11.9	10.4
500	8.1	14.7	12.6
600	8.9	15.5	13.2
700	9.3	15.7	18.6
800	11.8	21.7	23.9
900	19.5	30.0	25.0
1000	42.8	35.7	25.1
2000	59.0	50.1	35.8



(d) Alert Numbers

	Suricata	Snort	Bro
Traffic Rate	Alert	Alert	Alert
PPS	#	#	#
50	1973	1824	1963
100	1973	1824	1963
200	1973	1824	1963
300	1970	1739	1878
400	1970	1736	1865
500	1859	1695	1853
600	1854	1694	1843
700	1853	1694	1864
800	1847	1670	1853
900	1832	1602	1841
1000	1810	1593	1835
2000	1806	1502	1740

(e) Average Alert Numbers

IDS	#
Suricata	1893
Snort	1700
Bro	1872

Figure 5.7 Results of ICMP Attack Traffic

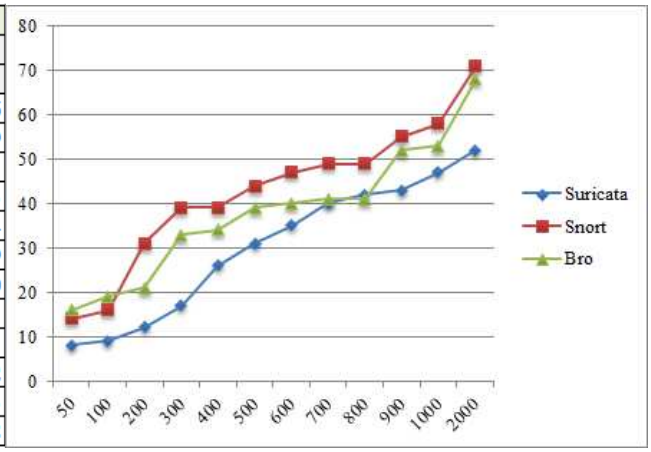
Figure 5.7 (a) shows the similar average memory usage among all three IDS for ICMP attack where Suricata has the lowest memory usage. Figure 5.7 (b) shows that the CPU usage has the increasing trends where Snort gives the highest trend and Bro gives the lowest trend. Figure 5.7 (c) presents the similar increasing trend of the number of packet loss where Bro seems to have the lowest packet loss but Suricata has the sharp increase in packet loss. Figure 5.7 (d) and (e) show the similar number of alerts for all three IDS where Suricata gives the highest number of alerts. Note that the number of alerts in ICMP attack is doubled from all previous types of attacks.

(a) Memory Usage

IDS	%
Suricata	17.2
Snort	20.3
Bro	18.3

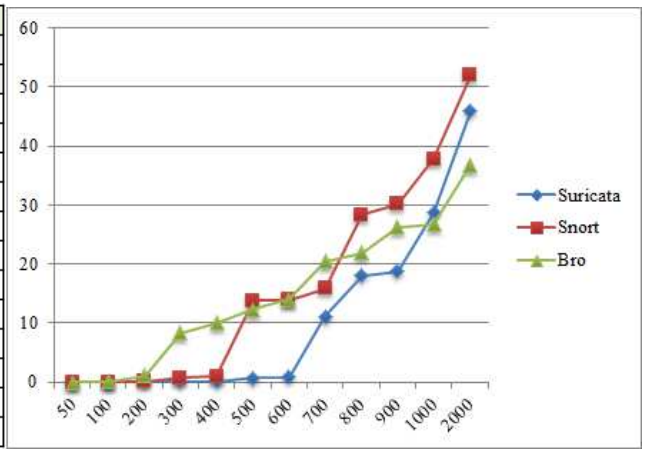
(b) CPU Usage

	Suricata	Snort	Bro
Traffic Rate	CPU	CPU	CPU
PPS	%	%	%
50	8	14	16
100	9	16	19
200	12	31	21
300	17	39	33
400	26	39	34
500	31	44	39
600	35	47	40
700	40	49	41
800	42	49	41
900	43	55	52
1000	47	58	53
2000	52	71	68



(c) Packet Loss

	Suricata	Snort	Bro
Traffic Rate	Packets Loss		
PPS	%	%	%
50	0.0	0.0	0.0
100	0.0	0.0	0.0
200	0.0	0.1	1.0
300	0.1	0.7	8.4
400	0.1	0.9	10.0
500	0.7	13.8	12.3
600	0.8	13.9	14.0
700	11.1	15.8	20.5
800	18.0	28.4	21.9
900	18.7	30.2	26.3
1000	28.7	37.8	26.7
2000	45.9	51.9	36.8



(d) Alert Numbers

	Suricat	snort	Bro
Traffic Rate	Alert	Alert	Alert
PPS	#	#	#
50	530	451	462
100	530	451	462
200	526	453	459
300	526	439	450
400	526	412	450
500	519	387	441
600	519	385	438
700	518	376	435
800	504	359	436
900	493	355	430
1000	450	346	421
2000	417	334	424

(e) Average Alert Numbers

IDS	#
Suricata	505
Snort	396
Bro	442

Figure 5.8 Results of POP3/IMAP Attack Traffic

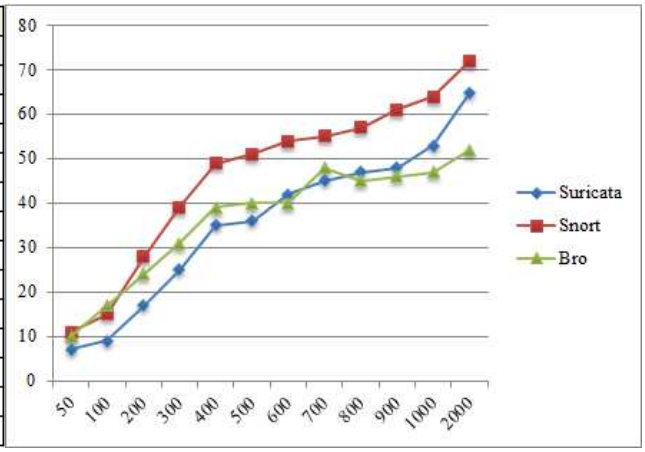
Figure 5.8 (a) shows the similar average memory usage among all three IDS for POP3/IMAP attack. Figure 5.8 (b) shows that the CPU usage has the similar increasing trends where Snort gives the highest trend and Suricata gives the lowest trend. Figure 5.8 (c) presents the similar increasing trend of the number of packet loss where Bro gives the lowest packet loss but Snort gives the highest packet loss. Figure 5.8 (d) and (e) show the similar number of alerts for all three IDS where Suricata gives the highest average number of alerts. Note that the number of alerts in POP3/IMAP attack is the lowest among all previous types of attacks.

(a) Memory Usage

IDS	%
Suricata	17.3
Snort	20.3
Bro	18.4

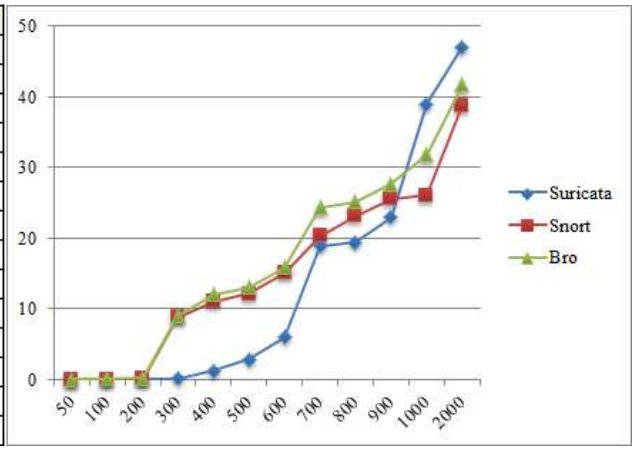
(b) CPU Usage

	Suricata	Snort	Bro
Traffic Rate	CPU	CPU	CPU
PPS	%	%	%
50	7	11	10
100	9	15	17
200	17	28	24
300	25	39	31
400	35	49	39
500	36	51	40
600	42	54	40
700	45	55	48
800	47	57	45
900	48	61	46
1000	53	64	47
2000	65	72	52



(c) Packet Loss

	Suricata	Snort	Bro
Traffic Rate	Packets Loss		
PPS	%	%	%
50	0.0	0.0	0.0
100	0.0	0.0	0.0
200	0.0	0.1	0.2
300	0.1	8.7	8.9
400	1.2	11.0	12.0
500	2.8	12.1	13.0
600	5.9	15.0	15.8
700	18.9	20.3	24.3
800	19.4	23.1	25.1
900	22.9	25.5	27.6
1000	38.9	26.0	31.8
2000	47.0	38.7	41.7



(d) Alert Numbers

	Suricata	Snort	Bro
Traffic Rate	Alert	Alert	Alert
PPS	#	#	#
50	1389	1129	1521
100	1389	1129	1521
200	1382	1103	1493
300	1362	1048	1392
400	1342	1029	1345
500	1294	1007	1330
600	1293	992	1327
700	1254	960	1319
800	1250	935	1397
900	1190	932	1389
1000	1102	922	1370
2000	1051	903	1291

(e) Average Alert Numbers

IDS	#
Suricata	1275
Snort	1007
Bro	1391

Figure 5.9 Results of SCAN Attack Traffic

Figure 5.9 (a) shows the similar average memory usage among all three IDS for SCAN attack. Figure 5.9 (b) shows that the CPU usage has the similar increasing trends where Snort gives the highest trend. Figure 5.9 (c) presents the similar increasing trend of the number of packet loss where Snort and Bro gives similar packet loss but Suricata has the sharp increase in packet loss at the end. Figure 5.9 (d) and (e) show the similar number of alerts for all three IDS where Bro gives the highest average number of alerts.

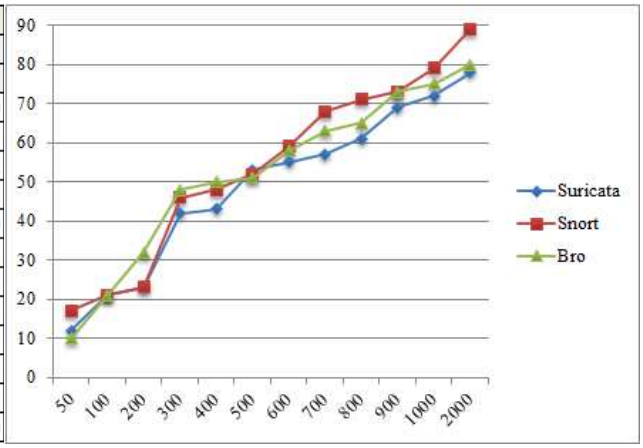
Figure 5.10 (a) shows the similar average memory usage among all three IDS for SNMP attack. Figure 5.10 (b) shows that the CPU usage has the similar increasing trends where Snort seems to be the highest among them. Figure 5.10 (c) presents the similar increasing trend of the number of packet loss. Figure 5.10 (d) and (e) show the similar number of alerts for all three IDS where Bro and Suricata give the highest average number of alerts.

(a) Memory Usage

IDS	%
Suricata	17.2
Snort	20.3
Bro	18.4

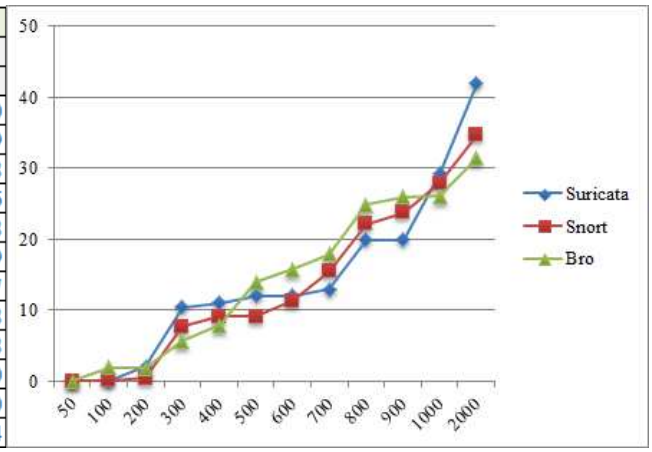
(b) CPU Usage

	Suricata	Snort	Bro
Traffic Rate	CPU	CPU	CPU
PPS	%	%	%
50	12	17	10
100	21	21	21
200	23	23	32
300	42	46	48
400	43	48	50
500	53	52	51
600	55	59	58
700	57	68	63
800	61	71	65
900	69	73	73
1000	72	79	75
2000	78	89	80



(c) Packet Loss

	Suricata	Snort	Bro
Traffic Rate	Packets Loss		
PPS	%	%	%
50	0.0	0.0	0.0
100	0.0	0.0	1.9
200	2.0	0.4	1.8
300	10.4	7.6	5.6
400	11.0	9.0	7.8
500	12.0	9.0	13.9
600	12.0	11.3	15.7
700	12.9	15.4	17.8
800	19.8	22.1	24.8
900	19.9	23.7	26.0
1000	29.2	27.8	26.0
2000	41.9	34.6	31.4



(d) Alert Numbers

	Suricata	Snort	Bro
Traffic Rate	Alert	Alert	Alert
PPS	#	#	#
50	992	901	1031
100	992	900	1023
200	963	892	1012
300	920	882	963
400	916	870	942
500	913	868	910
600	901	859	902
700	902	839	901
800	894	831	839
900	891	827	831
1000	890	810	827
2000	873	806	818

(e) Average Alert Numbers

IDS	#
Suricata	921
Snort	857
Bro	917

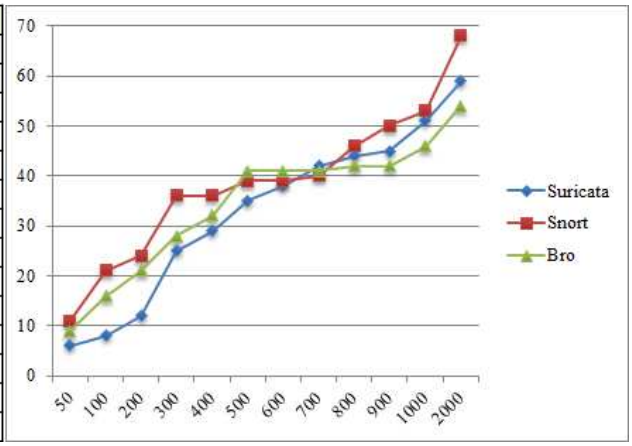
Figure 5.10 Results of SNMP Attack Traffic

(a) Memory Usage

IDS	%
Suricata	17.4
Snort	20.4
Bro	18.4

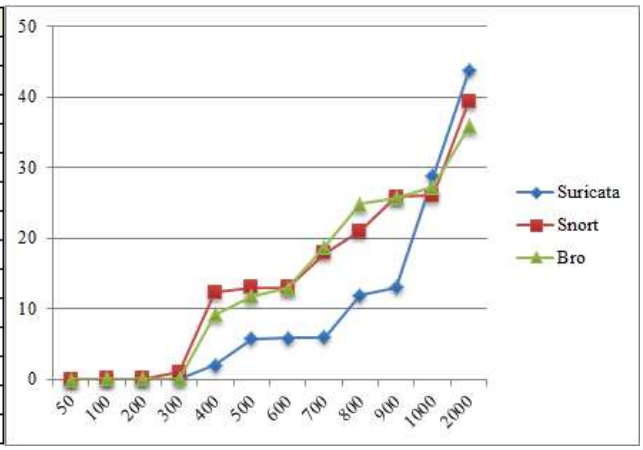
(b) CPU Usage

	Suricata	Snort	Bro
Traffic Rate	CPU	CPU	CPU
PPS	%	%	%
50	6	11	9
100	8	21	16
200	12	24	21
300	25	36	28
400	29	36	32
500	35	39	41
600	38	39	41
700	42	40	41
800	44	46	42
900	45	50	42
1000	51	53	46
2000	59	68	54



(c) Packet Loss

	Suricata	Snort	Bro
Traffic Rate	Packets Loss		
PPS	%	%	%
50	0.0	0.0	0.0
100	0.0	0.1	0.0
200	0.0	0.1	0.0
300	0.1	1.0	0.1
400	2.0	12.3	9.2
500	5.8	13.0	11.8
600	5.8	13.0	12.9
700	5.9	17.8	18.7
800	11.9	21.0	24.8
900	13.0	25.9	25.7
1000	28.7	26.0	27.3
2000	43.8	39.4	35.8



(d) Alert Numbers

	Suricata	Snort	Bro
Traffic Rate	Alert	Alert	Alert
PPS	#	#	#
50	918	830	993
100	918	829	992
200	915	813	990
300	901	782	984
400	893	751	969
500	881	750	973
600	880	743	971
700	873	737	954
800	869	731	937
900	852	720	932
1000	841	721	929
2000	810	716	901

(e) Average Alert Numbers

IDS	#
Suricata	879
Snort	760
Bro	960

Figure 5.11 Results of Telnet/SSH Attack Traffic

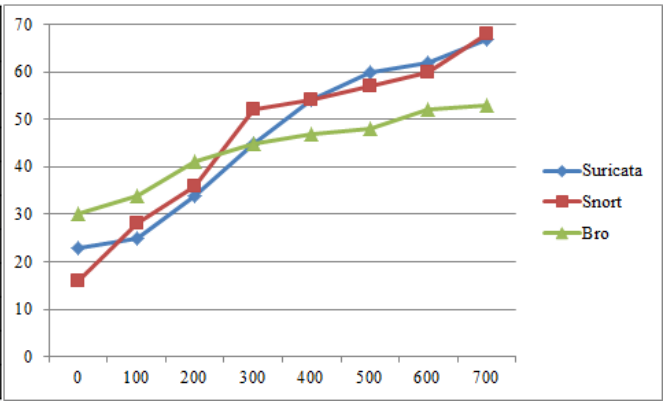
Figure 5.11 (a) shows the similar average memory usage among all three IDS for Telnet/SSH attack. Figure 5.11 (b) shows that the CPU usage has the similar increasing trends where Snort seems to be the highest among them. Figure 5.11 (c) presents the similar increasing trend of the number of packet loss for Snort and Bro whereas Suricata gives less number of packet loss till having sharp increase at the end. Figure 5.11 (d) and (e) show the similar number of alerts for all three IDS where Bro give the highest average number of alerts and Snort give the lowest average number of alerts.

5.4.3 Combined Traffic

This experiment shows the results when analyzing the combined traffic of normal and malicious traffic. The normal TCP traffic is the background traffic and is fixed at the rate 300 pps. For the malicious traffic, three attacks: DoS/DDoS attack, SCAN attack and SNMP attack are selected since they are very popular among IDS communities. In addition, only three performance metrics are chosen and they are the CPU usage, the packet loss and the number of alerts.

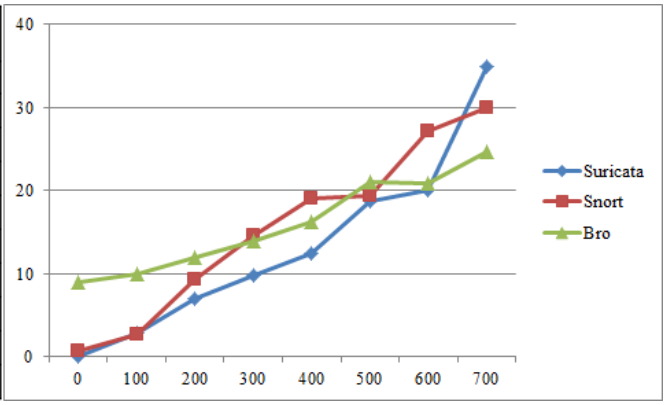
(a) CPU Usage

Malicious Traffic	Suricata	Snort	Bro
DoS/DDoS	CPU	CPU	CPU
(PPS)	%	%	%
0	23	16	30
100	25	28	34
200	34	36	41
300	45	52	45
400	54	54	47
500	60	57	48
600	62	60	52
700	67	68	53



(b) Packet Loss

Malicious Traffic	Suricata	Snort	Bro
DoS/DDoS	Packets Loss		
(PPS)	%	%	%
0	0	0.7	9
100	2.9	2.6	9.9
200	6.9	9.2	11.9
300	9.8	14.5	13.9
400	12.5	19	16.2
500	18.7	19.3	21.1
600	20.1	27.1	20.9
700	34.9	30	24.6



(c) Alert Numbers

Malicious Traffic	Suricata	Snort	Bro
DoS/DDoS	Alert	Alert	Alert
(PPS)	#	#	#
0	0	0	0
100	315	216	246
200	529	345	338
300	634	418	476
400	652	497	535
500	697	530	627
600	712	581	649
700	748	603	681

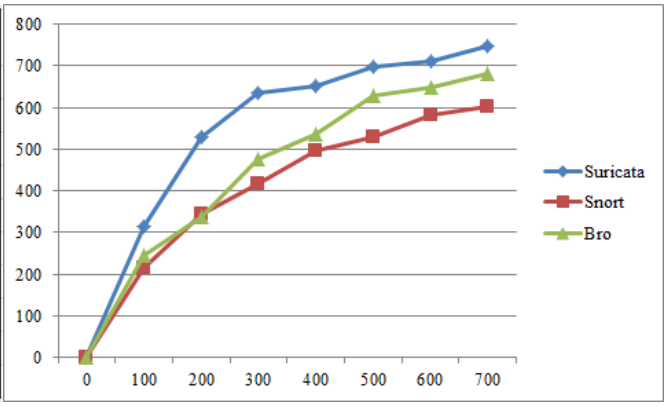
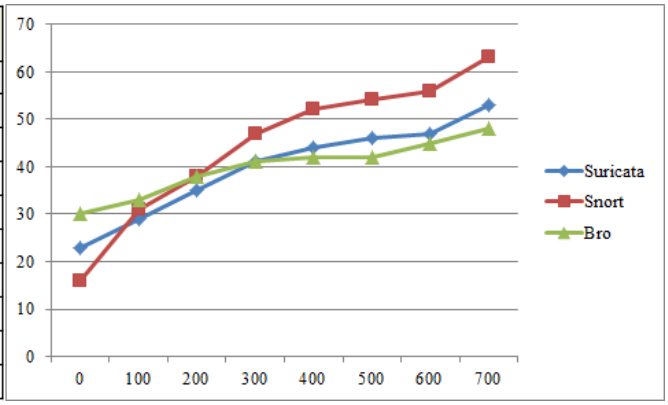


Figure 5.12 Results of Combined Normal and DoS/DDoS Traffic

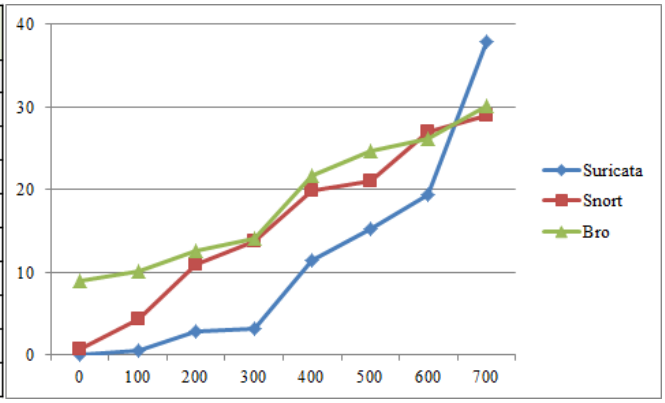
(a) CPU Usage

Malicious Traffic	Suricata	Snort	Bro
SCAN	CPU	CPU	CPU
(PPS)	%	%	%
0	23	16	30
100	29	31	33
200	35	38	38
300	41	47	41
400	44	52	42
500	46	54	42
600	47	56	45
700	53	63	48



(b) Packet Loss

Malicious Traffic	Suricata	Snort	Bro
SCAN	Packets Loss		
(PPS)	%	%	%
0	0	0.7	9
100	0.5	4.3	10.1
200	2.9	11	12.6
300	3.1	13.7	14.1
400	11.5	19.9	21.7
500	15.3	21.1	24.6
600	19.3	27	26.1
700	37.9	29	30.1



(c) Alert Numbers

Malicious Traffic	Suricata	Snort	Bro
SCAN	Alert	Alert	Alert
(PPS)	#	#	#
0	0	0	0
100	351	272	391
200	580	461	570
300	688	594	687
400	781	667	762
500	825	701	831
600	873	721	856
700	905	790	906

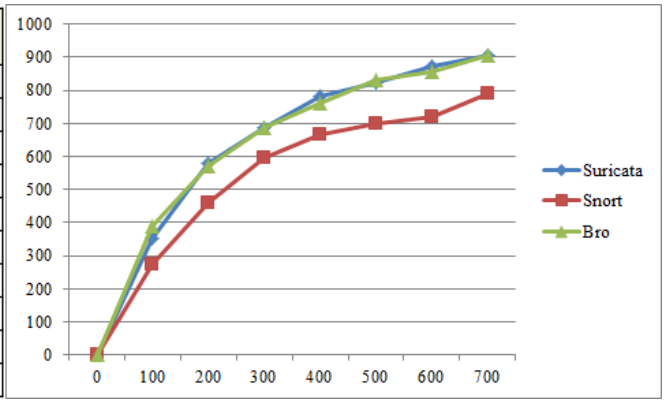
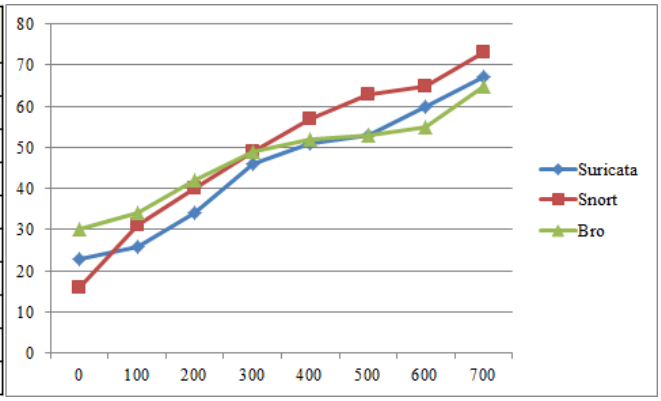


Figure 5.13 Results of Combined Normal and SCAN Traffic

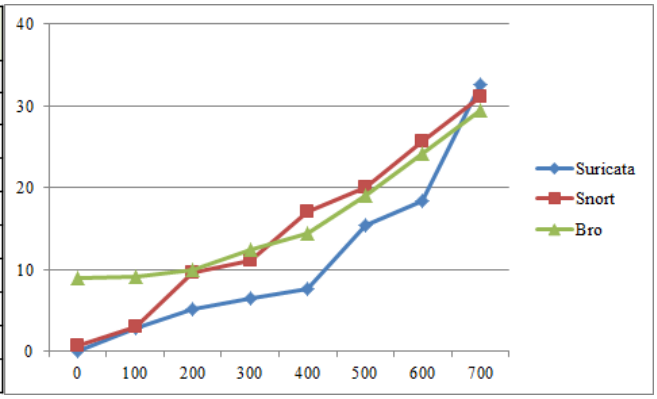
(a) CPU Usage

Malicious Traffic	Suricata	Snort	Bro
SNMP	CPU	CPU	CPU
(PPS)	%	%	%
0	23	16	30
100	26	31	34
200	34	40	42
300	46	49	49
400	51	57	52
500	53	63	53
600	60	65	55
700	67	73	65



(b) Packet Loss

Malicious Traffic	Suricata	Snort	Bro
SNMP	Packets Loss		
(PPS)	%	%	%
0	0	0.7	9
100	2.8	3	9.1
200	5.1	9.6	9.9
300	6.4	11.1	12.5
400	7.7	17	14.4
500	15.4	20.1	19
600	18.3	25.7	24.1
700	32.6	31.1	29.4



(c) Alert Numbers

Malicious Traffic	Suricata	Snort	Bro
SNMP	Alert	Alert	Alert
(PPS)	#	#	#
0	0	0	0
100	246	219	267
200	377	357	410
300	492	451	503
400	530	509	538
500	584	547	597
600	600	583	616
700	753	670	699

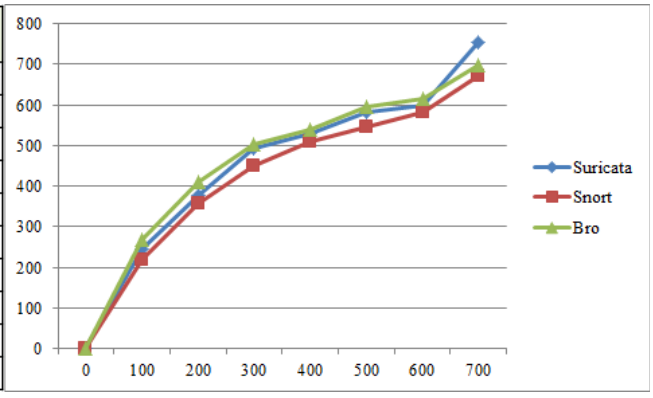


Figure 5.14 Results of Combined Normal and SNMP Traffic

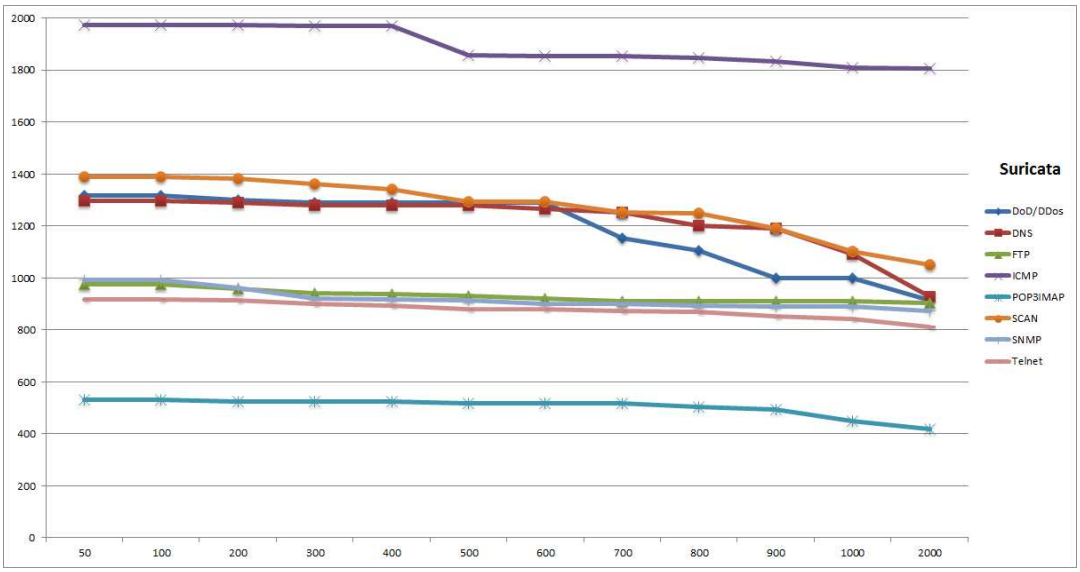


Figure 5.15 Alerts Number of Suricata in All Attacks

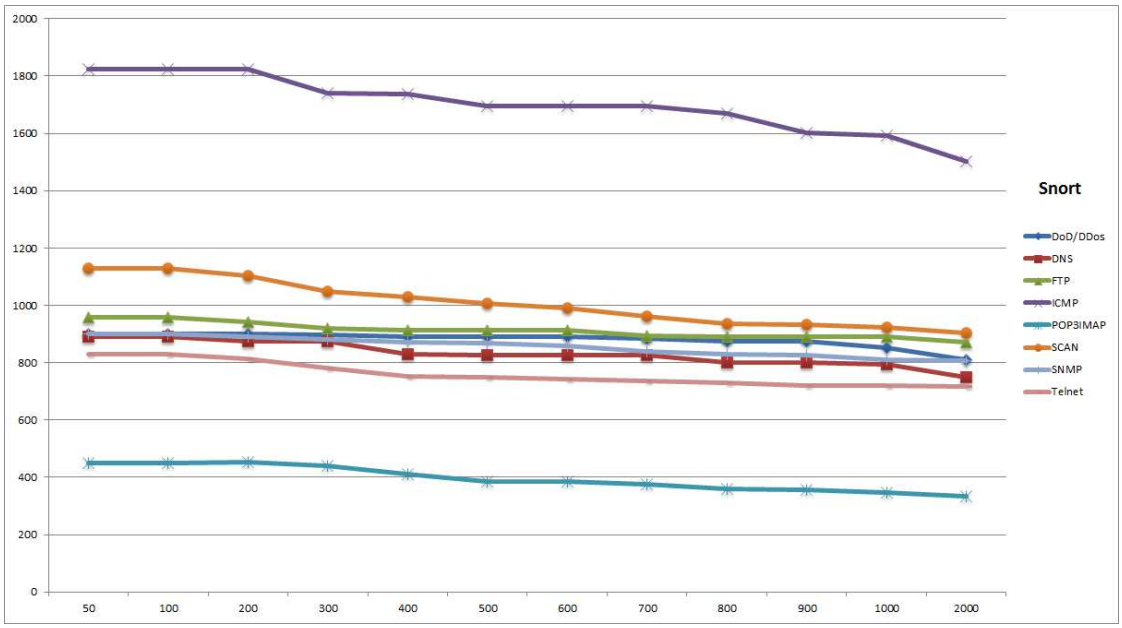


Figure 5.16 Alerts Number of Snort in All Attacks

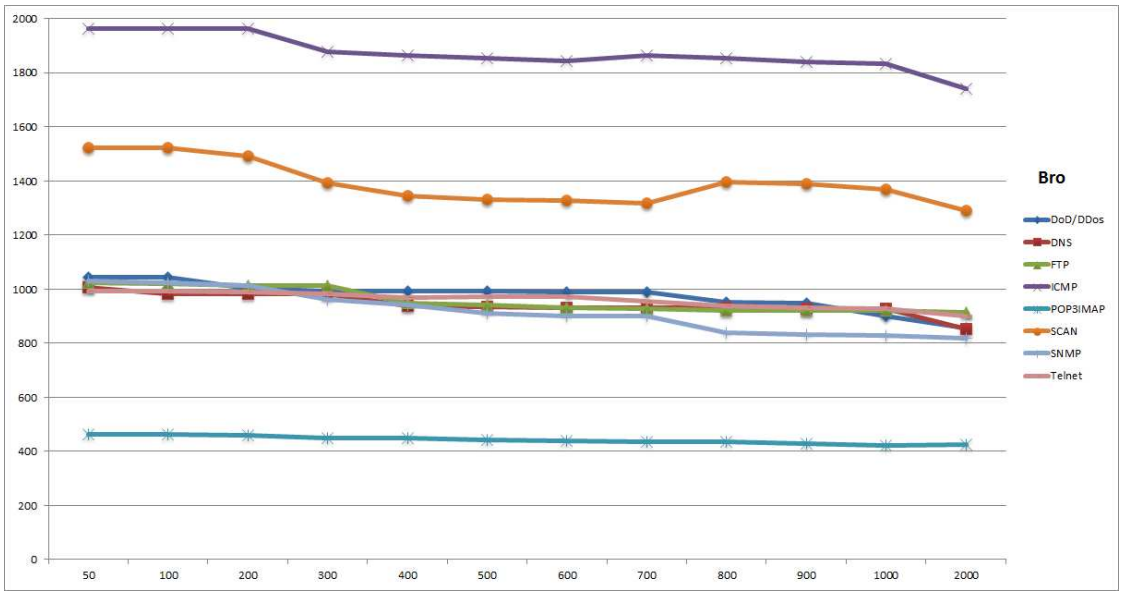


Figure 5.17 Alerts Number of Bro in All Attacks

Figures 5.15 to 5.17 compare the number of alerts in the eight attack types: DNS attack, DoS/DDoS attack, Ftp attack, ICMP attack, POP3/IMAP attack, SCAN attack, SNMP attack, and Telnet/SSH attack.

5.5 Experiment 2 and Results

This experiment measures the accuracy rate of the three IDS. The ratio of missed alerts is calculated from the number of alerts of all shared rules subtracted by the number of alerts of one attack rule set. The percentage of missed alerts is computed from the following formula.

Let X =Number of alerts of all shared rules

Y = Number of alerts of one attack rule

Thus, Percentage of missed alerts = (X -Y)/Y *100

Accuracy Rate = 100 - Percentage of missed alerts

Table 5.3, 5.4 and 5.5 show the number of alerts and the accuracy rate of each IDS. Figure 5.18 shows the comparison of the accuracy rate of all IDS in the graph.

Table 5.3Number of Alerts and Accuracy Rate for Suricata IDS

Attack Type	Suricata		
	1 Rule Set	All Rule Sets	Accuracy Rate
DNS	1298	1396	92.45
DoSDDoS	1318	1773	65.48
FTP	974	1055	91.68
ICMP	1973	2402	78.26
POP3IMAP	530	574	91.70
SCAN	1389	1585	85.89
SNMP	992	1138	85.28
TelnetSSH	918	985	92.70

Table 5.4 Number of Alerts and Accuracy Rate for Snort IDS

Attack Type	Snort		
	1 Rule Set	All Rule Sets	Accuracy Rate
DNS	890	958	92.36
DoSDDoS	901	1214	65.26
FTP	957	1034	91.95
ICMP	1824	2210	78.84
POP3IMAP	451	490	91.35
SCAN	1129	1292	85.56
SNMP	901	1041	84.46
TelnetSSH	830	894	92.29

Table 5.5 Number of Alerts and Accuracy Rate for Bro IDS

Attack Type	Bro		
	1 Rule Set	All Rule Sets	Accuracy Rate
DNS	1005	1094	91.14
DoSDDoS	1045	1410	65.07
FTP	1023	1107	91.79
ICMP	1963	2395	77.99
POP3IMAP	462	506	90.48
SCAN	1521	1737	85.80
SNMP	1031	1193	84.29
TelnetSSH	993	1072	92.04

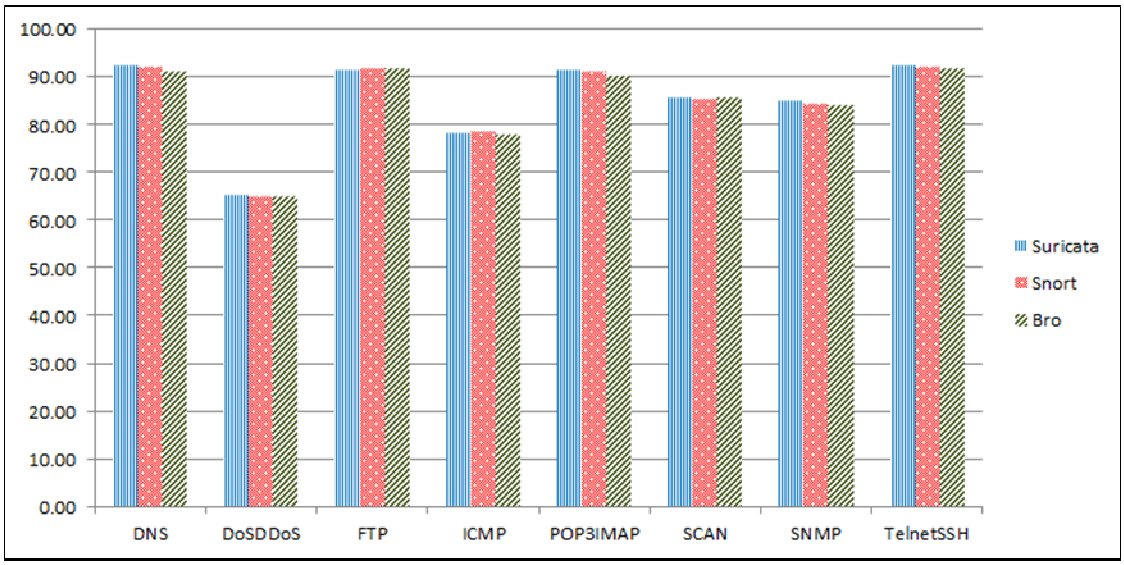


Figure 5.18 Accuracy Rateof Three IDS

5.6 Discussion of Experimental Results

The performance of each IDS is significantly affected by the increase of packet rates. From the experimental results previously presented, the analysis of the normal TCP and UDP traffic are shown in Figure 5.19 and all three IDS have different trends for the normal TCP and UDP traffic. In particular, Suricata has low packet loss for TCP traffic till the high packet rate whereas it has almost constant packet loss for UDP. However, both Bro and Snort has similar trends for both TCP and UDP traffic. In overall, the packet loss in UDP traffic is a little lower that those in TCP traffic.

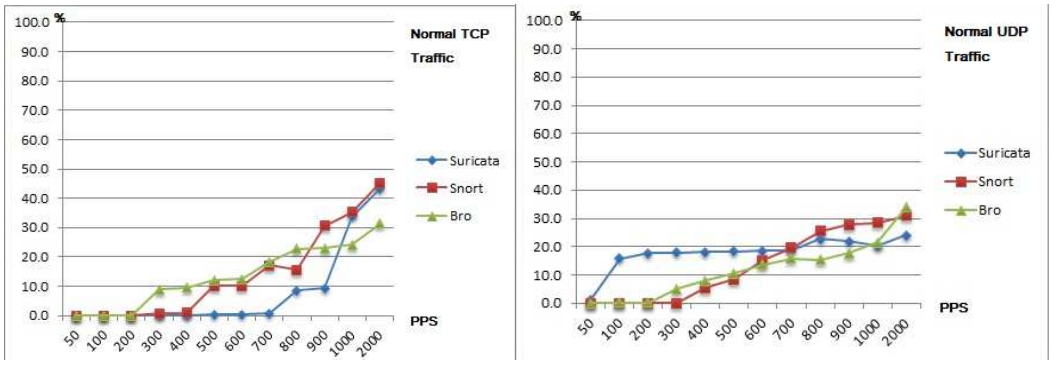


Figure 5.19 Packets Loss of TCP and UDP Normal Traffic

Figure 5.20 shows the CPU utilization of both the TCP and UDP normal traffic. When the three IDS have high packets loss, their CPU utilization is also increased as well since the packet rate affects the IDS performance. Even though Suricata gives lower packet loss, Bro gives lower CPU utilization. Moreover, the UDP traffic has the higher CPU utilization than the TCP traffic. In particular, Suricata has the worst CPU utilization for the UDP traffic.

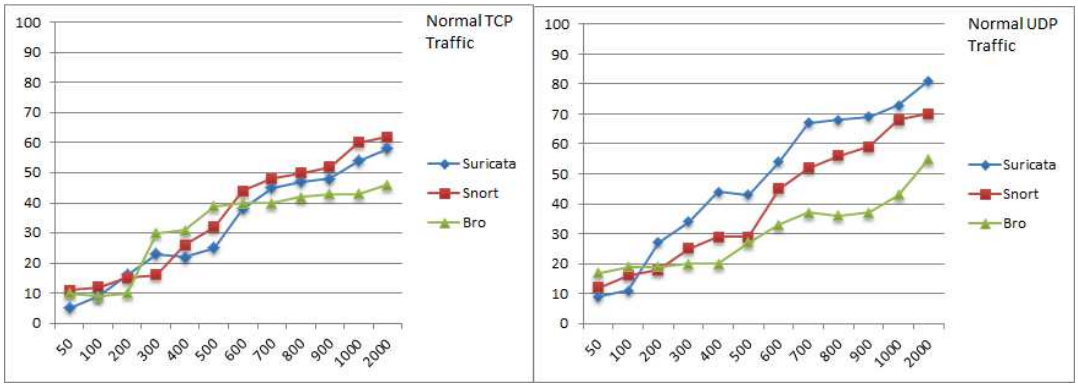


Figure 5.20 CPU Utilization of TCP and UDP Normal Traffic

Under the different kind of attacks, each IDS has different packet loss and CPU utilization. Table 5.6 shows the percentage of packet loss for all kinds of traffic, and they are plotted as graphs in Figure 5.21. In addition, Table 5.7 shows the percentage of CPU utilization for all kinds of traffic, and they are plotted as graphs in Figure 5.22.

The results show that Snort and Suricata use the CPU more than Bro. Figure 5.23also depicts that the memory usage of all three IDS has no much differences as the percentage is in between 17 to 21 %.

Table 5.6 Percentage of Packet Loss

IDS	Suricata			Snort			Bro		
Packet rate	200	400	700	200	400	700	200	400	700
TCP	0.0	0.0	0.7	0.0	0.9	17.1	0.0	9.6	18.3
UDP	0.1	18.0	18.7	0.0	5.4	19.7	0.1	8.0	15.9
DNS	2.6	9.0	18.2	1.3	15.1	18.4	2.0	11.4	19.8
Dos/DDoS	8.7	11.6	21.3	1.0	6.7	20.5	1.0	12.4	15.4
FTP	2.8	12.5	17.5	1.1	9.0	20.0	4.5	7.3	18.6
ICMP	0.0	0.1	9.3	0.1	11.9	15.7	0.1	10.4	18.6
POP3/IMAP	0.0	0.1	11.1	0.1	0.9	15.8	1.0	10.0	20.5
SCAN	0.0	1.2	18.9	0.1	11.0	20.3	0.2	12.0	24.3
SNMP	2.0	11.0	12.9	0.4	9.0	15.4	1.8	7.8	17.8
Telnet/SSH	0.0	2.0	5.9	0.1	12.3	17.8	0.0	9.2	18.7

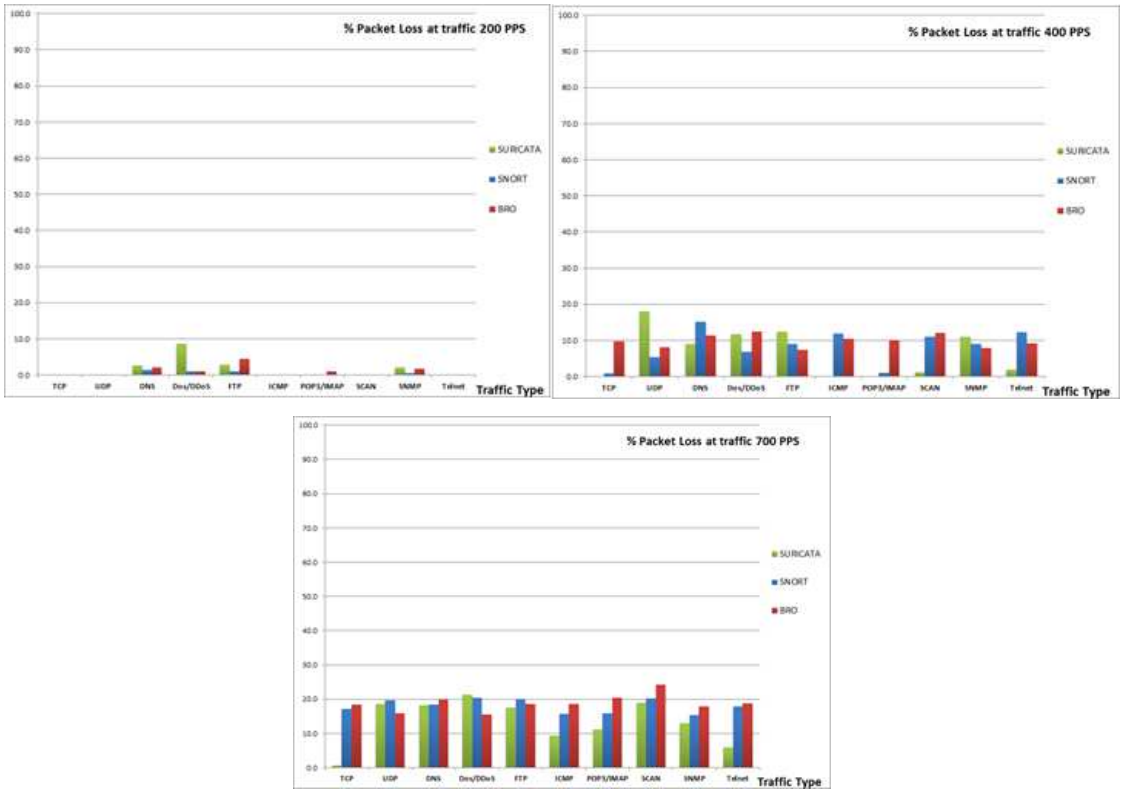


Figure 5.21 Percentage of PacketLoss

Table 5.7 Percentage of CPU Utilization

IDS	**Suricata**			**Snort**			**Bro**		
Packet rate	200	400	700	200	400	700	200	400	700
TCP	16.0	22.0	45.0	15.0	26.0	48.0	0.0	9.6	18.3
UDP	0.0	44.0	67.0	18.0	29.0	52.0	0.1	8.0	15.9
DNS	19.0	37.0	45.0	33.0	43.0	51.0	2.0	11.4	19.8
Dos/DDoS	22.0	36.0	60.0	41.0	45.0	62.0	1.0	12.4	15.4
FTP	19.0	42.0	51.0	29.0	45.0	53.0	4.5	7.3	18.6
ICMP	19.0	34.0	45.0	20.0	40.0	45.0	0.1	10.4	18.6
POP3/IMAP	12.0	26.0	40.0	31.0	39.0	49.0	1.0	10.0	20.5
SCAN	17.0	35.0	45.0	28.0	49.0	55.0	0.2	12.0	24.3
SNMP	23.0	43.0	57.0	23.0	48.0	68.0	1.8	7.8	17.8
Telnet/SSH	12.0	29.0	42.0	24.0	36.0	40.0	0.0	9.2	18.7

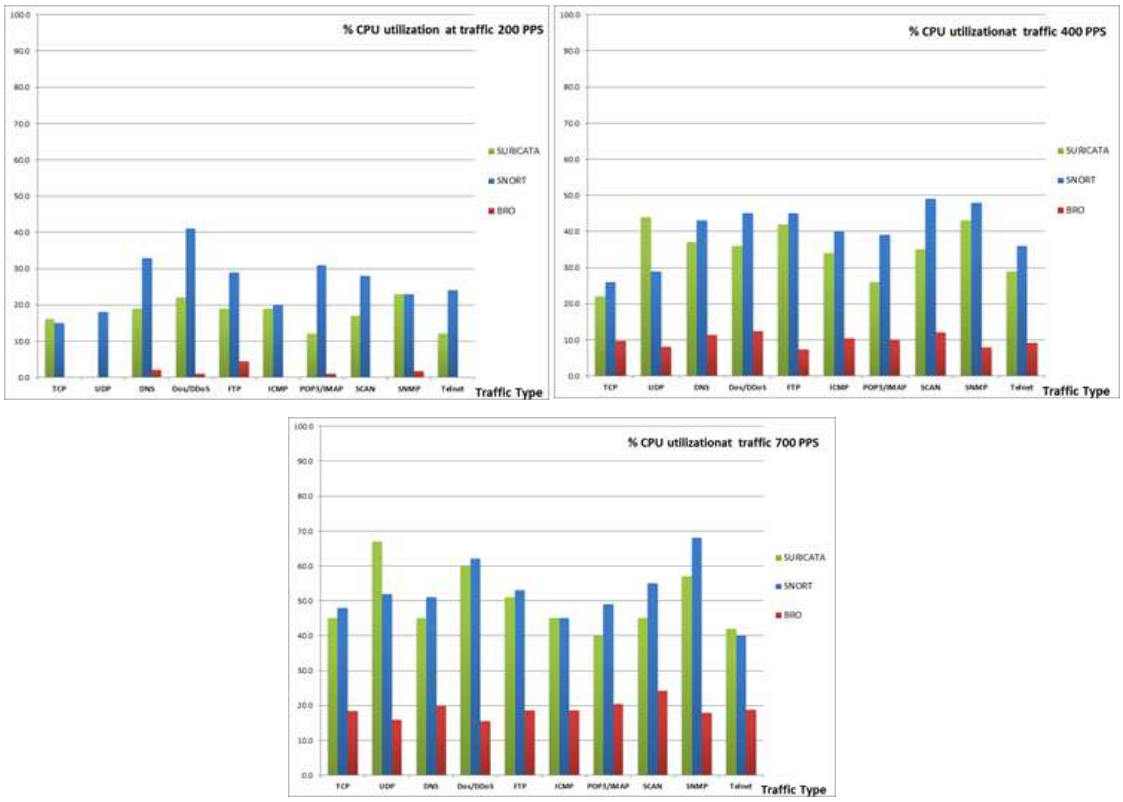


Figure 5.22 Percentage of CPU Utilization

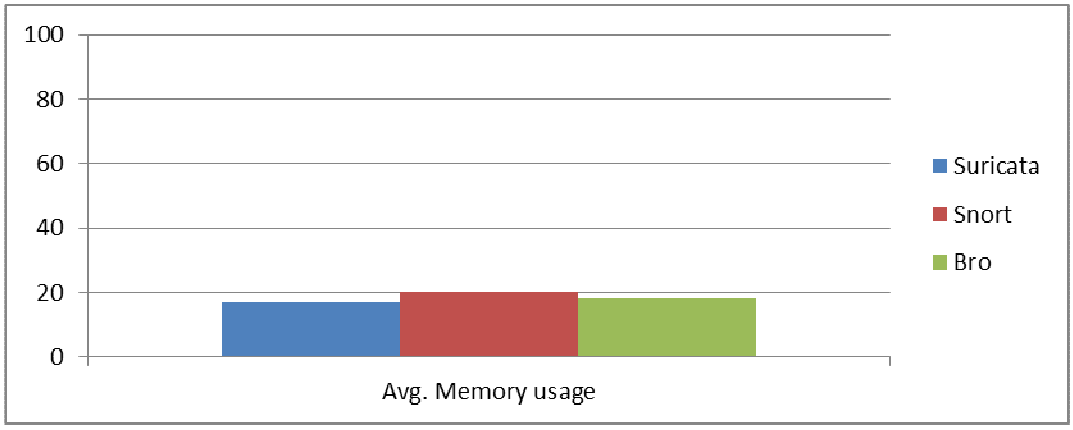


Figure 5.23 Average Percentage of Memory Usage

Figure 5.24 shows the results of the combined traffic of normal and DoD/DDoS attack traffic for all three IDS. At the low packet rate, Bro has higher packet loss and higher CPU utilization than other IDS, but after the rate of 200 pps, it has lower packet loss and lower CPU utilization than the others. However, Suricata gives the highest number of alerts. In addition, Figure 5.25 presents the packet loss and the CPU utilization for the packet rate of 200, 400,and 700 pps of all three IDS.

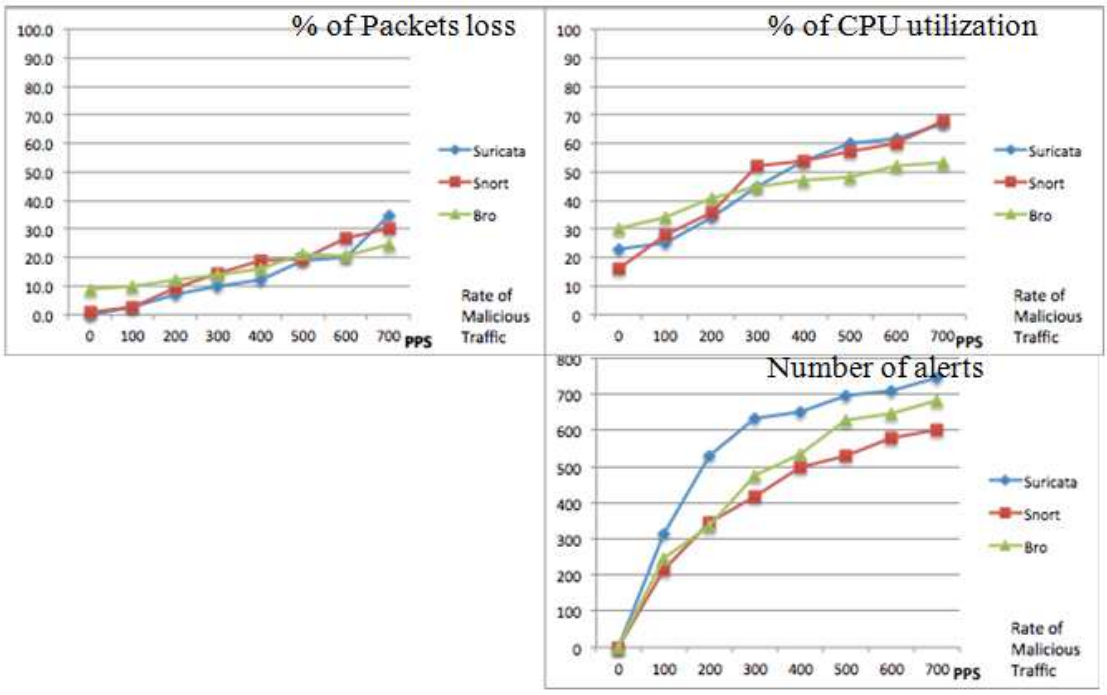


Figure 5.24 Combined Traffic of Normal and DoD/DDoS Attack Traffic

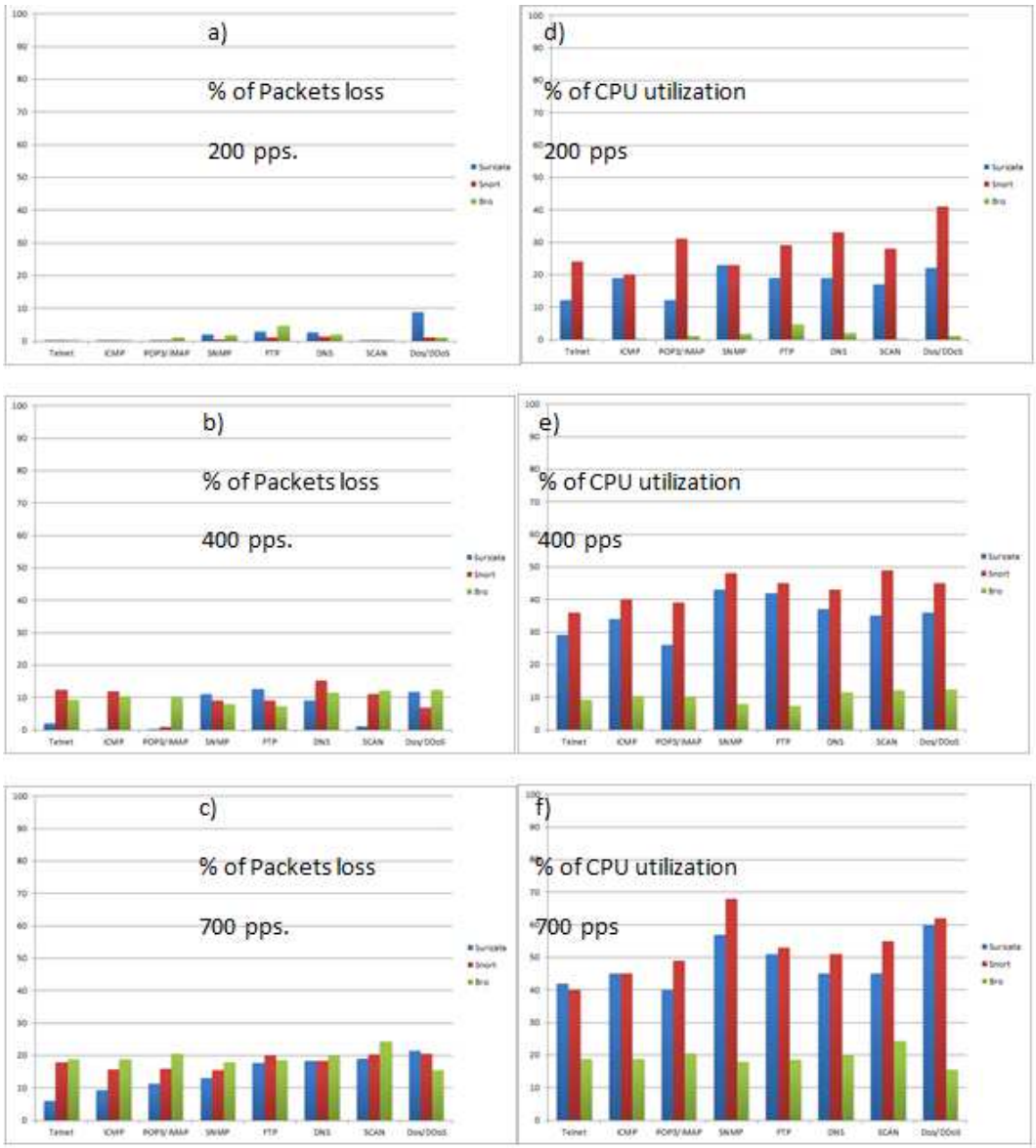


Figure 5.25 Packets loss and CPU utilization of 200, 400, and 700 pps.

CHAPTER VI
CONCLUSIONS

This chapter discusses the contribution of this thesis, discussed the problems and the limitations of our work that we found, and finally suggests the future work.

6.1 Contribution of the Thesis

This work analyzed and compared the three IDS tools: Snort Bro and Suricata in many perspectives. They are all different and similar in terms of the system architecture and the main components such as network traffic sensors, packet analyzers, data stores, response units, and especially the rule syntax. Our objectives are to evaluate all three IDS in terms of the performance and the accuracy. Thus, we set up the testing environment for evaluation.

The parameters we have considered include the traffic types to be normal and malicious traffic, eight selected attack types: DNS attack, DoS/DDoS attack, FTP attack, ICMP attack, POP3/IMAP attack, SCAN attack, SNMP attack, and Telnet/SSH attack, and a set of rules to be active: either a full set of rules or a specific set of rules. In the evaluation, we measure the performance in terms of packets loss and resource utilization (CPU and memory utilization). For the accuracy, we count the number of alerts when different set of rules are active.

From the experimental results, we summarize our findings as follows.

- Snort and Suricata gave similar results in terms of the performance and the accuracy, but it seems that Suricata performs a little bit better than Snort. This may be due to the new architectural design of Suricata that has been greatly improved.

- Similar to other studies, Bro still gave the best performance among three IDS.
- In our work, we used the same set of traffic for conducting the experiments for all three IDS. Thus, we could say that we compare the three IDS fairly.

6.2 Problems and Limitations

Our work has the following limitations.

- Our work is limited by the software used to generate both the background and the attack traffic in the testing environment since the generated background traffic and the percentage of mixing both traffic may not represent the real traffic.
- Our tests are limited by the type of attacks generated by NMAP. Thus, only eight attack types are chosen for this study. It would be beneficial if we can find a software tool that can generate many varieties of attacks since several attack types are challenging and interesting depending on the environment of the experiments.
- Another limitation is the compatibility of the rules defined in the IDS engine. Especially, in Bro, the rule syntax and the mechanism of the detection engine are defined differently. Thus, the results obtained from each IDS may not fully compatible.
- Since the size of the network traffic generated for testing was very large, it took very long time to run and do the analysis. However, the computer machines we used do not have high CPU power and large memory. Thus, the results of packet loss may not actually reflect the performance of IDS, but they give us the indication of how well the machines can handle high traffic volume.

6.3 Future Work

The evaluation of different IDS software is a challenging task since many IDS tools and techniques have been constantly developed. In the meantime, the new types of attacks and malicious traffic have also been evolved as well. Thus, the experiments and the desirable parameters should be frequently conducted to evaluate the performance and the accuracy of such IDS software so that the new research ideas would be introduced to help us battle out those malicious attacks efficiently.

CHAPTER VI
CONCLUSIONS

This chapter discusses the contribution of this thesis, discussed the problems and the limitations of our work that we found, and finally suggests the future work.

6.1 Contribution of the Thesis

This work analyzed and compared the three IDS tools: Snort Bro and Suricata in many perspectives. They are all different and similar in terms of the system architecture and the main components such as network traffic sensors, packet analyzers, data stores, response units, and especially the rule syntax. Our objectives are to evaluate all three IDS in terms of the performance and the accuracy. Thus, we set up the testing environment for evaluation.

The parameters we have considered include the traffic types to be normal and malicious traffic, eight selected attack types: DNS attack, DoS/DDoS attack, FTP attack, ICMP attack, POP3/IMAP attack, SCAN attack, SNMP attack, and Telnet/SSH attack, and a set of rules to be active: either a full set of rules or a specific set of rules. In the evaluation, we measure the performance in terms of packets loss and resource utilization (CPU and memory utilization). For the accuracy, we count the number of alerts when different set of rules are active.

From the experimental results, we summarize our findings as follows.

- Snort and Suricata gave similar results in terms of the performance and the accuracy, but it seems that Suricata performs a little bit better than Snort. This may be due to the new architectural design of Suricata that has been greatly improved.

- Similar to other studies, Bro still gave the best performance among three IDS.
- In our work, we used the same set of traffic for conducting the experiments for all three IDS. Thus, we could say that we compare the three IDS fairly.

6.2 Problems and Limitations

Our work has the following limitations.

- Our work is limited by the software used to generate both the background and the attack traffic in the testing environment since the generated background traffic and the percentage of mixing both traffic may not represent the real traffic.
- Our tests are limited by the type of attacks generated by NMAP. Thus, only eight attack types are chosen for this study. It would be beneficial if we can find a software tool that can generate many varieties of attacks since several attack types are challenging and interesting depending on the environment of the experiments.
- Another limitation is the compatibility of the rules defined in the IDS engine. Especially, in Bro, the rule syntax and the mechanism of the detection engine are defined differently. Thus, the results obtained from each IDS may not fully compatible.
- Since the size of the network traffic generated for testing was very large, it took very long time to run and do the analysis. However, the computer machines we used do not have high CPU power and large memory. Thus, the results of packet loss may not actually reflect the performance of IDS, but they give us the indication of how well the machines can handle high traffic volume.

6.3 Future Work

The evaluation of different IDS software is a challenging task since many IDS tools and techniques have been constantly developed. In the meantime, the new types of attacks and malicious traffic have also been evolved as well. Thus, the experiments and the desirable parameters should be frequently conducted to evaluate the performance and the accuracy of such IDS software so that the new research ideas would be introduced to help us battle out those malicious attacks efficiently.

REFERENCES

1. A. Alhomouda, R. Munira,J. P. Dissoa,I. Awana, b,A. Al-Dhelaan. Performance Evaluation Study of Intrusion Detection System. Procedia Computer Science 5 (2011) 173–180, 2011.
2. A. Ashoor and S.Gore, Importance of Intrusion Detection System (IDS) , IJSER, ISS 2229-5518, Jan. 2011.
3. Bro-IDS, http://www.bro-ids.org/
4. C. Kahn P. Porras S. Staniford and B. Tung, The Common Intrusion Detection Framework - Data Formats. IETF, 1998.
5. High Orbit Ion Cannon (HOIC), http://www.symantec.com
6. J. Mikians, P. Barlet-Ros, J. Sanjuàs-Cuxart, J. Solé-Pareta, A Practical Approach To Portscan Detection In Very High-Speed Links, Proc. Passive and Active Measurement Conference, 2011
7. K. Salah, A. Kahtani. Improving Snort performance under Linux. IET Commun, 2009, Vol. 3, Iss. 12, pages. 1883–1895, 2009.
8. K. Salah, A. Kahtani. Performance Evaluation Comparison of Snort NIDS under Linux and Windows Server. Journal of Network and Computer Applications 33 (2010) pages 6–15, 2010
9. Low Orbit Ion Cannon (LOIC), http://sourceforge.net/projects/loic/
10. M. Akhlaq, F. Alserhani, I. Awan, J. Mellor, A. J. Cullen, and A. Al-Dhelaan, Implementation and Evaluation of Network Intrusion Detection Systems. LNCS 5233, pp. 988–1016, 2011.
11. Miguel A. CalvoMoya, Analysis and evaluation of the Snort and Bro network intrusion detection system,Universidad PontificiaComillas, 2008.
12. N. Paulauskas, J. Skudutis. Investigation of the Intrusion Detection System “Snort” Performance. ISSN 1392-1215 No.7(87), 2008.
13. Network Mapper(NMAP), http://nmap.org/
14. OSSEC, http://www.ossec.net/
15. Ostinato, http://code.google.com/p/ostinato/
16. Prelude, http://www.prelude-technologies.com/

17. P. Proctor, The practical Intrusion Detection Handbook. Prentice Hall, 2001.
18. Russ McRee, August 2010, “Suricata: An Introduction, http:// holisticinfosec.org/ toolsmith/pdf/august2010.pdf
19. The Common Intrusion Detection Framework (CIDF), http://gost.isi.edu/cidf/
20. Snort, http://www.snort.org/
21. S. Ngamsuriyaroj, B. Sa-nguankwamdee, E. Maharattanaviroj and P.Ua-sopon, Measurement of Snort Performance under Various Attacks.
22. S. Staniford-Chen, B. Tung, P. Porras, C. Kahn, D. Schnackenberg, R. Feiertag, and M. Stillman, The Common Intrusion Detection Framework - Data Formats. IETF, 1998.
23. Suricata, http://www.openinfosecfoundation.org/
24. V. Gowadia,C. Farkas, and M. Valtorta., Paid: A probabilistic agent-based ntrusiondetection system. In Computers & Security, volume 24, pages 529–545, 2005.

BIOGRAPHY

NAME	Kittikhun Thongkanchorn
DATE OF BIRTH	6February1985
PLACE OF BIRTH	KhonKaen, Thailand
INSTITUTIONS ATTENDED	Mahidol University, 2004-2007
	Bachelor of Science (Computer Science)
HOME ADDRESS	89/562 Bangtey, Sampran, Nakhonpathom 73210
EMPLOYMENT ADDRESS	Faculty of ICT, Mahidol University ,
	Rama 6 Road, Rajathevi, Bangkok 10400
	E-mail :kittikhun.tho@mahidol.ac.th