



**SUDAN UNIVERSITY FOR SCIENCE AND TECHNOLOGY**



**College of Graduate Studies**

**College of Computer Science and Technology**

**Hybrid Based Network Intrusions Detection Systems**

**الأنظمة الهجينة لإكتشاف الدخلاء في الشبكة**

**Fulfillment of the requirements of Master of Science degree in Computer Science**

**By: Malik Khalil Mohamed Alfaki**

**Supervision of**

**Dr. Mohammed Al-Ghazali Hamza Khalil**

**November 2018**

# **DECLARATION**

I hereby declare that the work reported in this M.Sc. thesis titled as “Approach for Anomaly-based Intrusion Detection System using SNORT” submitted at the Faculty of Computer Science / Sudan University of Science and Technology, is an authentic record of my work carried out under the supervision Dr. Mohammed Al-Ghazali Hamza. I have not submitted this work elsewhere for any other degree.

## **Dedication**

To my family special to who I carry his name with pride (my father) and to the greatest woman that I have met that always support me (my mother) who are support me to complete the educational road, all of you are my strength and thanks for everything. I would like to extend my sincere thanks to my colleagues, lecturers and the others for contributing and supporting me directly and indirectly. Thanks for your support, comments and advice. Finally, thank you to all and who commitment in making this project successful.

## **ACKNOWLEDGMENT**

I wish to express my deepest appreciation to all those who helped me, in one way or another, to complete this project. First and foremost I thank God almighty who provided me with strength, direction and purpose throughout the project. Special thanks to my project supervisor Associate Professor **Dr. Mohammed Al-Ghazali Hamza** for all his patience, guidance and support during the execution of this project. Through her expert guidance, I was able to overcome all the obstacles that I encountered in these enduring ten months of my project. In fact, she always gave me immense hope every time I consulted with her over problems relating to my project.

## **ABSTRACT**

Intrusions detection systems (IDSs) are systems that try to detect attacks as they occur or after the attacks took place. IDSs collect network traffic information from some point on the network or computer system and then use this information to secure the network. The Intruder is someone without permission or privilege try to access to system resource performs unnecessary/unauthorized activities, intrusion detection means detecting unauthorized activity or attacks upon system or network. An IDS is a system that generates an alarm when suspicious activity or use on system or network when it configured its monitor the gateway or host to identify the intruder. Two type of detect method: anomaly detection (also called behaviour-based) and signature-based (also named misuse or pattern based), signature-based detection identify the intruder by matching according to predefined signature of attack, if signature match is classified as an attack, but it cannot detect the attack that is not stored in the signature database which cannot detect novel attack (zero-day attack). Anomaly detection can detect both the known and novel attack due to that anomaly detection based on the preserve profile for normal activity of system or network if any deviation is considered as an attack. This research provides a review of various Intrusion Detection Systems and its tools by focusing on SNORT IDS-an open source tool, also provide an extension for SNORT by adding pre-processor to detect anomalies, that make the system hybrid based detection which increase the rate of intruder detection.

أنظمة الكشف عن التطفل (IDS). هي أنظمة تحاول اكتشاف الهجمات عند حدوثها أو بعد وقوع الهجمات. تقوم IDS بجمع معلومات حركة مرور الشبكة من نقطة ما على الشبكة أو نظام الكمبيوتر ثم تستخدم هذه المعلومات لتأمين الشبكة ؛ يعرف الشخص الدخيل أو الهاكر في علم الحاسوب بأنة هو شخص بدون إذن أو امتياز يقوم بمحاولة الوصول إلى موارد النظام ويقوم بتنفيذ أنشطة غير ضرورية / غير مصرح بها ، نظام كشف التسلل (IDS) هو عبارة عن نظام يقوم بكشف اي نشاط غير مصرح به أو هجمات على نظام أو شبكة. إن نظام (IDS) يقوم بتوليد التنبيه عند وجود نشاط مريب أو اي سوء استخدام على النظام أو الشبكة ، نوعان من طرق الكشف: الكشف عن الشذوذ (يطلق عليه أيضاً نظام مراقبة السلوك للكشف عن سوء الاستخدام) ، و النوع الثاني يعتمد على النمط و الادلة (ويسمى أيضاً النمط المعتمد) والاكتشاف القائم على النمط المعين الذي يحدد هوية الدخيل من خلال التطابق وفقاً لمنط الهجوم المحدد مسبقاً المخزن في قاعدة بيانات ، إذا تم تطابق النمط يصنف على أنه هجوم لكن هذه الطريقة لا يمكنها الكشف عن الهجوم الذي لم يتم تخزينه في قاعدة بيانات .و بالتالي انه لا يمكنه الكشف عن هجوم جديد الذي يعرف بـ (zero day attack) .. أما الطريقة الثانية الكشف عن الشذوذ يتم التحديد على أساس حفظ الملف للنشاط الطبيعي للنظام أو الشبكة إذا كان أي انحراف عن هذا النشاط يعتبر هجوم و احتمالية وقوع هجوم او حدوث دخول غير مصرح به. يقدم هذا البحث نظرة عامة لمختلف أنظمة كشف التسلل وأدواتها من خلال التركيز على أداة مفتوحة المصدر تعرف بـ (SNORT) ، كما يقترح بتقديم امتداداً عن طريق إضافة المعالج لنظام (SNORT) للكشف عن الحالات الشاذة ، مما يجعل النظام هجيناً قائم علي دمج طريقتي الكشف في نظام واحد و الذي يزيد معدل الكشف عن الدخلاء أو الهاكر.

## List of Figures

<b>Figure</b>		<b>Page</b>
2.1	Host IDS	9
2.2	Network IDS	10
2.3	IDS Before Gateway Firewall	11
2.4	IDS Within De-Militarized Zone (DMZ)	11
2.5	IDS inside the private Network	11
3.1	SNORT components	20
3.2	BASE interface	22
3.3	Proposed New System Architecture	24
3.4	System process view	26
3.5	CommView – Main page	27
4.1	Snort Sniffer mode	31
4.2	Sniffed Packet Detail	31
4.3	DDOS (TFN) ICMP possible communication	35
4.4	BASE alert of ICMP traffic	35
4.5	BASE Alert (DDOS ICMP possible communication)	36
4.6	BASE Alert (DDOS ICMP possible communication)	37
4.7	BASE Alert ( large UDP Packet)	37
4.8	BASE Alert ( BAD TRAFFIC TCP, UDP port 0 traffic)	38
4.9	BASE Alert ( BAD TRAFFIC data in TCP SYN packet)	39
4.10	BASE Alert (BAD TRAFFIC same SRC/DST IP address)	40
4.11	BASE Alert (BAD TRAFFIC loopback traffic)	41
4.12	Snort Result number of Packets Processed	43
4.13	Number of Alerts for Darpa Signature test	44

4.14	Time Chart vs Number of Alert (Signature Detection test)	44
4.15	BASE Alerts about DARPA Evaluation (Signature Detection test)	45
4.16	Alert Category from DARPA Evaluation (Signature Detection test)	46
4.17	number of alert per-day (Signature Detection test)	46
4.18	Time of alert per-hour (Signature Detection test)	47
4.19	The result of Snort with SPADE number of Packets Processed	48
4.20	Number of Alerts for anomaly data test	49
4.21	Time Chart vs Number of Alert per-day (Anomaly Detection test)	49
4.22	Alert Categorization vs number of alerts (Anomaly Detection test)	50
4.23	BASE Alerts about DARPA Evaluation on anomaly pre-processor	50
4.24	number of alert per-day (Anomaly Detection test)	51
4.25	Time of alert per-hour (Anomaly Detection test)	52



## LIST OF ABBREVIATIONS

**IDS:** Intrusion Detection System. A system that detects any intruder activity. SNORT is example of IDS.

**HIDS:** Host Intrusion Detection System, a system that detects an intruder in a single host

**NIDS:** Network Intrusion Detection System, a system that detects intruder for the network usually single device.

**SNORT:** Free Software develops for Intrusion Detection.

**Alert:** A message generated when any intruder activity is detected. Alerts may be sent in many different forms, e.g., pop-up window, logging to screen, e-mail and so on.

**Snort Configuration File:** The snort.conf file, which is the main configuration file for Snort. It is read at the time when Snort starts.

**Snort Rule:** A way of conveying intruder signatures to Snort.

**Signature:** A pattern that corresponds to a known threat.

**Anomaly:** abnormal activity or deviation from normal activity.

**Preprocessors:** one of snort component for performing some operation for the packet which is rearranged or modified before the packet goes to the detection engine.

**ICMP:** internet control message protocol network protocol used to echo and replay the message

**Flow:** A particular network communication session occurring between hosts.

**IDPS:** intrusion detection/ prevention system

# TABLE OF CONTENTS

TABLE OF CONTENTS.....	
CHAPTER ONE.....	1
INTRODUCTION.....	1
Preface.....	1
1.1 Brief history of ids.....	3
1.1.1 Types of ids.....	4
1.2 Problem statement.....	5
1.3 Objectives.....	6
1.4 Organization of thesis.....	7
CHAPTER TWO.....	8
LIT REVIEW.....	8
1. INTRODUCTION.....	8
2.1 Intrusion Detection System Overview.....	8
2.1.1 Operation and Function of IDS.....	8
2.1.2 Classification of Intrusion Detection System.....	8
2.1.3 Limitation of Intrusion Detection System.....	10
2.1.4 Intrusion Detection Sensor Placement.....	10
2.1.5 IDS processing of the information.....	11
2.1.6 How to protect IDS itself.....	12
2.2 Intrusion Prevention System.....	13
2.3 Related Work.....	13
CHAPTER THREE.....	19
METHODOLOGY.....	19
Introduction.....	19
3. Research Methodologies and Tools to be adopted.....	19
3.1 Operating Environment.....	19
3.1.1 SNORT.....	20
3.1.2 BASE (Basic Analysis and Security Engine).....	21
3.1.3 ADODB.....	22
3.1.4 PHP.....	22
3.1.5 Virtual Box.....	22
3.2 Improving the current IDSs systems.....	23
3.3 Proposed New System Architecture.....	23
3.4 Statistical Packet Anomaly Detection Engine (SPADE) Preprocessor.....	24

3.4.1 The concept of Anomaly detection with SPADE.....	24
3.4.2 Function of SPADE.....	25
3.5 System Process View.....	25
3.6 Evaluating IDS.....	27
3.6.1 Using CommView to Generate Packet.....	27
3.6.2 DARPA Intrusion Detection System Data Set.....	28
3.7 Validation and Sureness.....	28
3.8 Result presentation.....	28
3.9 Summary.....	29
CHAPTER FOUR.....	30
Results and Discussions.....	30
Introduction.....	30
4.1 Operate With SNORT.....	30
4.1.1 Sniffer Mode.....	30
4.1.2 Network Intrusion Detection Mode.....	32
4.1.3 Working with rules.....	33
4.2 Implementation Part 1: Experimentation Signature Rule Set.....	33
4.2.1 Signature 1 : DDOS (TFN) ICMP possible communication.....	34
4.2.3 Signature 3: large UDP Packet.....	37
4.2.5 Signature 5: BAD TRAFFIC data in TCP SYN packet.....	38
4.2.6 Signature 6: BAD TRAFFIC same SRC/DST IP address.....	39
4.2.7 Signature 7: BAD TRAFFIC loopback traffic.....	40
4.3 Experimentation signature base detection Evaluation.....	42
4.4 Experimentation SPADE Anomaly base detection Evaluation SPADE preprocessor.....	47
5. Result and Discussion.....	52
CHAPTER FIVE.....	53
Conclusion & Recommendation.....	53
REFERENCE.....	55
Appendix A.....	59
Appendix B.....	66
Appendix C.....	67
Appendix D.....	76

# **Chapter One**

## **INTRODUCTION**

# CHAPTER ONE

## INTRODUCTION

### **Preface**

Nowadays with the spreading of the Internet and online procedures requesting a secure channel, it has become an inevitable requirement to provide network security. There are various threat sources including software bugs mostly as the operating systems and software used becomes more functional and larger in size.

Intruders who do not have rights to access these data can steal valuable and private information belonging to network users. Firewalls are hardware or software systems placed in between two or more computer networks to stop the committed attacks, by isolating these networks using the rules and policies determined for them. It is very clear that firewalls are not enough to secure a network completely because the attacks committed from outside of the network are stopped whereas inside attacks are not. This is the situation where intrusions detection systems (IDSs) are in charge. IDSs are used in order to stop attacks, recover from them with the minimum loss or analyze the security problems so that they are not repeated [1].

### **Background of research**

Computer Network is a collection or a set of the device the connected together which can be capable to exchange information and data. As the computer network was spread wide, the needing of the security becomes most important because lots of vulnerabilities and threats are come up on a computer network.

Due to this increase with the appearing of the Internet revolution, a lot

of problem and security issue have been arising to forcing security for the system and critical information across the network. Most companies and organization's use the internet to provide a channel of communication with their customer's and people to give them Availability of the service's, so while this information available for the harmless user's in the same hand this information available for malicious user's.

For protecting from the negative side of availability of information's most organization use or deploy Different traditional security way for protection against threat such as Firewalls, even in this case firewall not became safeguard against this threat for simple reason such as cannot analysis into packet to mining for the malicious activity or data to determine it harm or not, so the security issue is keeping arising, as result many technologies are created and developing to decrease this issue and provide protection for network while at the same time provide secure way to share information to outside via internet and networks.

Network security is the process of detection and prevention of computer information and resources, detection method help to detect who's not authorize access and prevention allow to avoid those suspect which known as an intruder.

Intrusion Detection System (IDS) is the software that monitors the network traffic and analysis packet to determine the malicious activity against security, the main objective of this software is to detect and inform about the suspect activity in the network or the system, The concept of Intrusion Detection / Prevention System (IDPS) is arising and gain popularity this software and tools enhance the concept of security level for organization and play important role in security in the depth method.

## **1.1 Brief history of ids**

The concept of detecting the intrusion's, system misuse and the intruders or looking for some kind of malicious activity and user's abnormal activity is discussed by James Anderson in his report titled "*Computer Security Threat Monitoring and Surveillance*" [1] to US Air Force in the year 1980.

In 1984 there was the first prototype of Intrusion Detection System which monitors the user activities "Intrusion Detection Expert System" (IDES), also in 1988 first IDS "Haystack" which depend on the patterns and statistical analysis for detecting malicious activities, but it has lacked in detecting in the real-time traffic and analysis.

In 1989 at University of California Davis' Lawrence Livermore Laboratories, they built IDS called "Network System Monitor" (NSM) for analyzing the network traffic, which developed to IDS named "Distributed Intrusion Detection System" (DIDS) and (Stalker) first IDS based on the (DIDS) which is commercially available.

In the 90's SAIC developed "Computer Misuse Detection System" (CMDS), a host-based IDS. The Intrusion Detection System began to gain popularity and market in 1997, in the same that year security market leader, ISS was developed IDS software which called "Real Secure". So many companies aware about the importance of security software which can be provided to the customer so the commercial IDS world expanded its market-based. Snort is was launched by Martin Roesch in 1998 which is an open source intrusion detection system and getting success in the market base and gain a good reputation, other IPS with is founded by Okena Systems in 1999 is the first intrusion

Prevention System which name “Strom Watch”, which IPS can add extra level of defense in the network by providing protection to it which co-operate with firewall. Host-based intrusion Detection Company, mCentrax Corporation, emerged as a result of a merger of the development staff from Haystack Labs and the departure of the CMDS team from SAIC.

### **1.1.1 Types of ids**

Depending on the needing of the IDS or the way that IDS will be deployed/ Implemented, it can be classified into two major types:

#### **Network Intrusion Detection System**

This type of IDS which Design to analysis and monitor each packet in the network and determine the abnormal and malicious activity in the packet this IDS designed to fill the gap by a firewall’s simplistic filtering rules.

#### **Host Intrusion Detection System**

This type of IDS configures and install in computer and host who monitor the device activity which includes all login attempts, process schedules, system files integrity checking system call tracing and kernel activity to find any abnormal activity or processes.

**DETECTION TECHNIQUE** There are two main approaches to detecting, which is signature-based detection and anomaly-based detection.

#### **Signature-based detection**

The type use matching techniques against a well-known attack signature or pattern, the sensor of IDS will be deployed to monitor and compare



the traffic if the packet match with library or database of known signatures of attack then the attack is detected [2]. This technique is useful to detect already known attacks but not the new ones if signature not in the database it will not detect the attack.

An advantage of Signature detection [3] does not only detect the intrusion it's also can detect the attempts of intrusion, but partial signature may also indicate an intrusion attempt.

### **Anomaly-based detection**

In this type is detect abnormal activity and behaviour in the system and deviation from normal behaviour is considered as abnormal activity, the detection depends on many measurement and metrics which may include traffic rate, number of packets for each protocol etc, is called normal profile, these profiles compare with any new activity in the system everything out of this profile is classified as malicious activity, the detection is done by heuristics or rules, rather than patterns or signatures[4].

## **1.2 Problem statement**

According to the above discussion intrusion can pose a serious security risk and choosing the appropriate security architecture and software's is the most of one important security goals for any company or organization, using these technologies and software provide level and layer of defences to fight existing threats and provide the ability to analysis data inside the network. Most intrusion detection system use one of these two ways misused detection or Anomaly detection, to secure the system's and both of them have their own limitations.

The pros and cons of these two techniques is that not better than each

other, the signature detection is able to detect the malicious activity if the pattern is matched, on the same hand is not able to detect to if the pattern is not recognized and stored in databases, the anomaly detection method is able to detect attack.

For this, it needs an intelligence solution or system that fill the gap, system that increase the detection rate of the malicious activity, which can detect based on signature detection method and anomaly detection combined together.

### **1.3 Objectives**

A reliable and efficient intrusion detection system (IDS) is one of the main and important components in any network. With this component is can alert administrators of possible attackers and provide a view of the network's status, the objective of this project is that:

- 1- To study appropriate security architecture for protection by examining the origins of detecting, analyzing, and reporting of malicious activity.
- 2- To create System of detection which by combines the two detection techniques together in one system by using SNORT tool
- 3- To provide a comparative study on the tool before and after adding the anomaly detection technique.

Network security importance was increased in last year's, cause many companies and home user's kept the sensitive information and data in a computer, so they are needing a method to protect their information form exploit it, one of this method is Intrusion Detection System (IDS). with simulating experimental studies in virtual environment to implement approach for detection method by combining the two techniques of detection (Anomaly, Signature) in one system, this project

is performed by implementing an open source signature-based IDS called SNORT, this software come with a complete package for Signature Detection, by adding anomaly Detection method make SNORT for effective defense against the Network attacks.

#### **1.4 Organization of thesis**

This research organize as follow, Chapter 1 contain an introduction to the research topic, Types and techniques for IDS, statement of problem, objective and scope, chapter 2 contain discusses published information, related work and theoretical focus that in the area of project and general information about system architecture, chapter 3 this section research methodology, inside this chapter research design and procedures were presented as well as equipment, Assumptions, chapter 4 which contain result and discussion about the implementation of work and finally the conclusion and future work at Chapter 5.

## **Chapter Two**

### **THEORETICAL FOCUS AND RELATED LITERATURE**

## **CHAPTER TWO**

### **LIT REVIEW**

#### **1. INTRODUCTION**

The focus of this chapter is to provide the theoretical base information for the study such as operation and function of IDS, classification, limitation and deployment and then related work in the field of IDS is also covered.

#### **2.1 Intrusion Detection System Overview**

The rapid growth of computer networks has changed the prospect of network security; up to this time, the researcher has been developing Intrusion Detection System (IDS) to detect a different type of attack and malicious activity in the different environment.

The idea behind have intrusion detection system is that you have a device that can see all the traffic that passes through the network, it all time looks on the traffic based on pre-defined detection rule, this system will generate an alert if there any match between the detection rule and incoming packet.

##### **2.1.1 Operation and Function of IDS**

An intrusion detection system is created to serve three major functions which is: monitor, detection and response to unauthorized activity in the company, which by that intrusion detection can use based rule policies match for detecting this activity with predefined security even then IDS send an alert if detect one of this event.

##### **2.1.2 Classification of Intrusion Detection System**

Intrusion detection system can be classified based on many factors the

position where the intrusion detection system takes place this can determine the type of IDS [5].

### **Host Intrusion Detection System**

Used to detect the malicious activity on the host or individual device like a server, which monitors the OS activity and application looking for any malicious movement, The function is a monitor only that device that installed inside, the decision-based host activity usually like (audit trails) and Logs files.

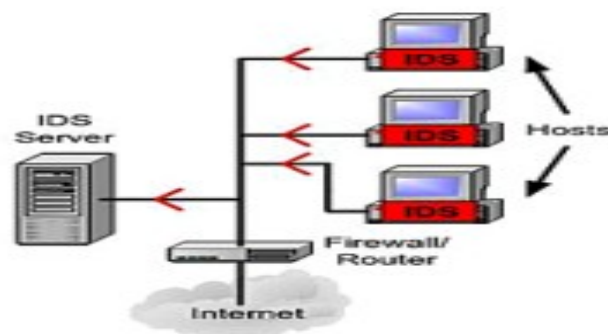


Figure 2.1: Host IDS

### **Network Intrusion Detection System**

This type operates with the network data flow traffic monitor all this traffic and looks for intruder packet that has been passed within network traffic, the process of monitor an entire network with a few sensors in key positions, normal this type of IDS set in the entry point of the network.

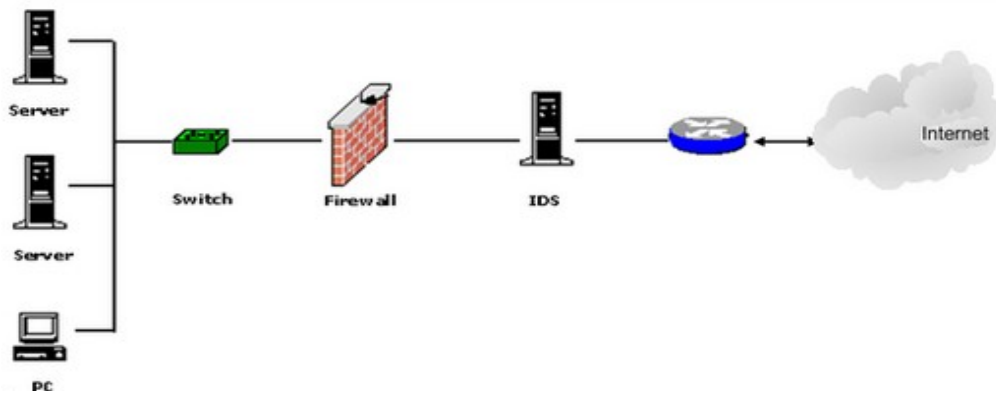


Figure 2.2: Network IDS

### 2.1.3 Limitation of Intrusion Detection System

The limit of Intrusion Detection Systems is protection which that IDS that they cannot preempt or prevent network attacks traffic because IDS sensors are based on packet sniffing (packet capture) technologies that only watch network traffic that pass through it cannot take any action to stop these traffic [6].

In the case of the preventing or stop the malicious traffic, it can be used Intrusion Prevention System, is that software used to protect the device or network form attacking by stop the packets of attack.

### 2.1.4 Intrusion Detection Sensor Placement

The most important roles in detection are the sensor which for effective monitoring of the network traffic, three positions can be used to placing the sensor in the system [6]:

1- Before the Gateway firewall: here the IDS will monitor and analysis all the traffic that passed through the network, which handles a large packet.

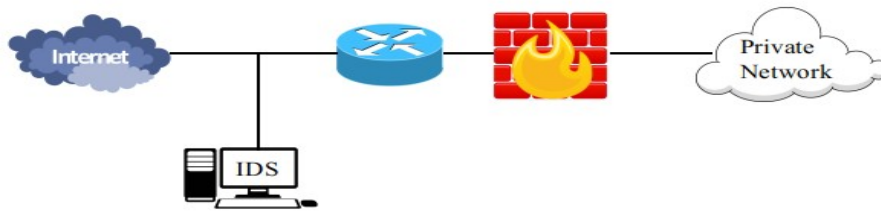


Figure 2.3: IDS before Gateway Firewall

2- Within De-Militarized Zone (DMZ): in this case, the IDS will monitor and analysis the packet that is filtered by a firewall, the work of IDS it reduced, due to firewall packet filter.

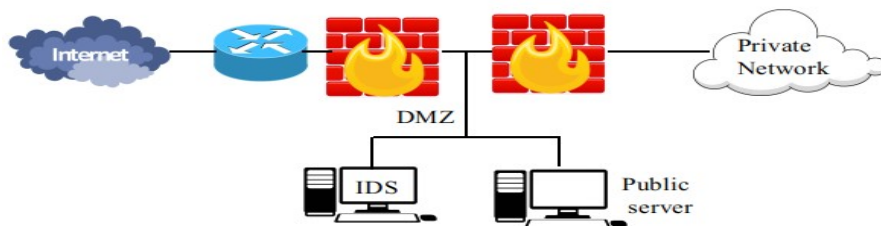


Figure 2.4: IDS within De-Militarized Zone (DMZ)

3- Finally IDS can be placed inside the network which can monitor the local traffic for malicious activity that from local area.

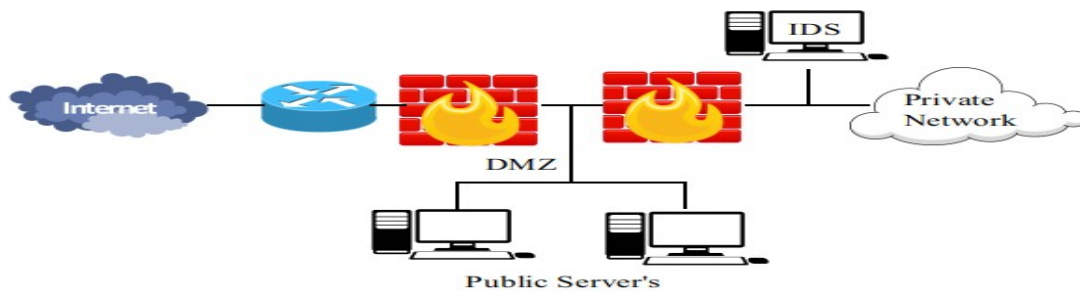


Figure 2.5: IDS Within Privide LAN

### 2.1.5 IDS processing of the information

When all information and data is required, the next step for IDS is to analysis that collected data by using analysis data engine to determine the malicious traffic or activity on the system and this done by:



**Misuse detection engine:** this detection base on the predefined knowledge <sup>[7]</sup> such as signature detection based on matching the signature to packet and activity and rule detection which based on the like define set of (if-then) rule for the detect attack [8].

**Stateful protocol analysis:** general predefined profile of protocol activity is set and compare with a new activity to define the deviation and classified as intrusion activity [9].

**Anomaly Detection:** the main objective of this type is to find anomalies and abnormal activity, the activity will be classified as attack activity, many measurements can be used for this approach some of them depend on the Statistical based methods, rule-based method, profile base method and Model-based methods; these methods will be discussed further in chapter 3.

### 2.1.6 How to protect IDS itself

IDS systems protect our system and our information form arising attacks, but there is important thing that must putted in mind is that how to protect the system that your IDS is running on, cause when your IDS is effect or compromised by attack in the case you will get a false alarm (wrong alert), there is many method and way's to protect your system of IDS form many recommendations, [10] provide some of it:

- It better to not run any service with IDS sensor, cause this service can be used as an entry to attack, simply can be easily exploited by an attacker.
- Always be up to date with you IDS software, for the simple reason is that most threats are discovered by the vendor, so it's better to get the last release which can contain new defense ways and method.

- Denial of service (DOS) most common attack that can stop your IDS system, so configure your IDS to not a response to any ICMP echo-type protocol
- You can use block techniques such as IPTABLES for protection.

## **2.2 Intrusion Prevention System**

An Intrusion Prevention System is a module added to the Intrusion Detection System. This module provides the ability to perform specific tasks automatically. Intrusion Prevention Systems perform the same analysis as Intrusion Detection Systems but, because they are inserted in-line, between other network components, they can block malicious activity, An IT administrator can define the actions to be taken by the IPS when the attack comes automatically disconnects and drop the packet's <sup>[11]</sup>.

Recently, Robert <sup>[12]</sup> states that Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS) respectively are complete network-level defences. The basic difference between the two technologies lies in how they provide protection for host and network environments, Intrusion Detection System analyzes network traffic and generates alerts when malicious activity is discovered and intrusion prevention system to protect against this malicious activity

## **2.3 Related Work**

Numerous research works are going on these days in IDS for better improvement in the performance and functionality Of Detection Systems. Intrusion detection system software like SNORT is based on signature pattern matching [13], matching pattern-based detection which is matching the network traffic for well-known attack signature, this technique is well with the all known attack type the negative side is that

it's vulnerable to novel attacks, so another approach is founded with is anomaly based detection this type based on the abnormal or the anomalies activity, the deviation from normal will be classified as suspicious activity.

In [14] discuss and implement the signature-based detection approach by using snort software, the IDS can be detected and analyze the real-time network traffic, Basic analysis and security engine (BASE) is also used to see the alerts generated by Snort, it's also mentioned that the advantages of signature IDS such as there are low false positive alert cause all attack signature is known and clearly defined, secondly is that the signature is easy to use and implement, in the disadvantage is discuss that the signature IDS is needed more knowledge about attack pattern before the signature be out of date, secondly it may not be practical inside attack involve abuse privilege

A detailed review in hybrid IDS is displayed in [15] presented much major research done in the Hybrid IDS. Much information is mentioned about IDS as general, for the performance of IDS many measurements are discussed as follows:

first false positive alert or event which is the IDS begin to define intrusion activity but this activity is none attack, second false negative this event happens when the IDS system is not able or fail to detect the intrusion/attack while it's, in fact, is a real attack.

After viewing many papers and research this paper conclude, that when it using anomaly-based detection will be able to detection unknown attack will be increased and the rate of the false positive and false negative, and if you use the signature-based detection method here will not be able to detect the unknown or the new attack, cause their

signature is not stored in the database.

Comparison of performance for differing hybrid IDS is discussed, the comparison is based here on the many measurements which are (AC = Accuracy, DR = Detection Rate, FP = False Positive, FA = False Negative).

With a general overview of various Intrusion Detection tool, [16] proposed a new approach for hybrid IDS by using snort which called H-SNORT which extend the capabilities of Snort. The enhance of the snort will be by the automatically generate patterns of misuse from attack data, the new pre-processor base on de-fragmentation of the package in the network to avoid evasive attacks in the IDS.

<sup>[17]</sup> Describe that the Intrusion Detection System is becoming necessary, provide the comparison of the Detection method, general presentation techniques of Intrusion (Detection and Prevention) System's, on the other side description of the evaluation, comparison and classification features of the IDS and the IPS.

Secondly, the characteristics of IDPS technologies and provides recommendations for designing, implementing, configuring, securing, monitoring, and maintaining them is discussed.

For the Host IDS it classifies the Host IDPS into two types:

- 1- The HIDS Based Application. The IDS of this type receive the data in the application
- 2- The HIDS Based Host. The IDS of this type receive the information of the activity of the supervised system.

For the second type of the IDS, it classifies the Network IDS into:

- 1- Passive IDS this type of IDS which can provide alert to admin

without take an action

2- Active IDS this type can react with the intrusion after detection it like (block, drop and stop)

Anomaly Detection IDS error classified into two types approach which

1- False positives (incorrectly identifying benign activity as malicious)

2- False negatives (failing to identify malicious activity).

<sup>[18]</sup> Provide general study about open source network intrusion detection software (SNORT), this study cloud help in the analyzing the performance of SNORT IDS through generate alert in high speed network traffic, also it mentioned that snort is the single-threaded application which has five mode: Packet Sniffer Mode, Packet logger mode, Detection mode, Prevention Mode/ Inline Mode include (Drop, Reject, drop).

For Detection Criteria of NIDS, the main function of snort is two: Packet Sniffing and Packet Logging. Snort is Signature-based detection so the rule in the snort is divided into main part or two logical section: Rule header, Rule Option.

Rule header contains packet information such as rule's action, protocol, source and destination IP addresses. Rule option is two common type in Snort (Alert and Logging).

The result of this experiment is found out the snort signature of attack and the number of sources that generate the attack and the destination for the attack also every signature have a unique ID, the result of that which is every alert provided with full detail and information of signature.

Comparative analysis of signature-based intrusion detection and

anomaly-based intrusion detection systems are discussed here <sup>[4]</sup>. Snort and PHAD Detection Systems are used in this here where is Snort is Signature-based Detection and PHAD is Anomaly-based detection, Snort is lightweight signature-based detection depend on the pattern match and compare the signature on the database to detect the attack, while the PHAD is the time-based protocol anomaly detector for network packets, to apply the time based protocol detection to anomaly the formula is became =  $TN/R$  , where is N the number of the packet is observed during the period, R the number of distinct values of a particular packet field observed during time and T is time since last anomaly detection.

For the evaluation of PHAD and SNORT in this experiment has been evaluating by using 1999 DARPA off-line IDS evaluation data set which contain the network traffic TCP-DUMP files. The period of this evaluation and measured by weeks: week3, week4 and week5. Week3 perform attack for training the PHAD on the detection and week4 and week5 is performing the real attack for the testing phase. Snort also evaluate and tested by DARPA Data-set.

The detection rate may be measured by this formula:  $\delta = d/n$  which is:

$\delta$  = Intrusion detection rate.

d = number of the detected attack.

n = number of the actual attack.

For PHAD the number of detected attack have been detected = 5, total number of data-set of actual attack = 201, so the detection rate for PHAD is  $\delta = 5/201 * 100 = 2.49\%$ .

For the SNORT no of detected attack = 116, detection rate:  $\delta = 116/201$

\* 100 =57.71 %.

So the compared analysis between the PHAD and SNORT is done with the following parameter: alarms generated protocol wise and detection rate. It has been found and concluded that Snort is better in performance than PHAD.

Intrusion detection and intrusion prevention system with most common free based software tool (SNORT) is discussed on [10] which mentioned that the main drawback of Network Intrusion Detection system that monitors traffic is that attacker can evade the detection by exploiting ambiguities in the traffic stream as seen by the NIDS [19].

According to for the Host IDS is have an advantage over Network IDS is that H-IDS can access the encrypted information when it's sending over the network for source to destination, in the side of detection techniques the Statistical Anomaly-Based Intrusion Detection System is taking place in this paper, in the way that detection work, profile is created from the normal activity this profile consist of metric data such as number of packets (traffic), the number of packet for each protocol [20].

In the same hand of detection techniques, Stateful Protocol Analysis Intrusion Detection Prevention System also mentioned and it concept that improvement of the packet filter concept by comparing the e packet, connection (Protocol) with pre-defined protocol profile to find deviation.

Many aspects are discussed also such as work with the snort rule, snort alert and other things. Finally, the experiment is successfully implemented and performed on Windows OS, attack of the specific signature is detected and alert is generated successfully by snort rules.

**Chapter Three**  
**METHODOLOGY**



## **CHAPTER THREE**

### **METHODOLOGY**

#### **Introduction**

This chapter demonstrates the research design and procedures were presented as well as instrumentation, and the explanation of the methodology to be followed to achieve the research objectives.

#### **3. Research Methodologies and Tools to be adopted**

For the implement the desired system architecture, many tools and techniques will be needed and used to perform many tasks, below here a summary of the tools and software that will be used for.

##### **3.1 Operating Environment**

This project implement of Linux platform with free open source software SNORT. The Operating system that will be used is Linux Centos Operating system with following software packet:

- SNORT (Network Intrusion Detection System).
- BASE (basic analysis and security engine).
- MySQL and ADODB.
- PHP.
- Apache2
- DARPA dataset
- Virtual Box (virtual machine player)
- Packet generator

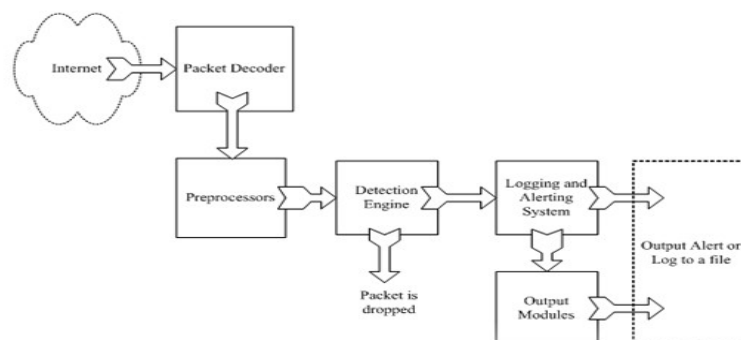
### 3.1.1 SNORT

Snort lightweight open source system became widely used in many company and organization cause it's flexibility and easy to use and configure, snort work as a packet sniffer which mean that it's capture every packet that transmits over the network and displays this packet in different level of detail such detail of network layer or data link layer. You can get it from snort website [21].

#### Components of SNORT

Snort is based on many components this component is cooperation together to detect an incoming attack and how to respond for this action; the components are five as follow:

1. Packet Decoder
2. Preprocessors
3. Detection Engine
4. Logging and Alerting System
5. Output Modules



**Figure 3.1: SNORT components**

All this component work together from packet decoder until the output module provides information about attack packet which used to make a decision either drop, log or generate an alert. The above components each one has a specific function to perform as follow [22].

## **Packet Decoder**

The network interface could be Ethernet, SLIP, or PPP... etc., the packet decoder collects these packets from the different network interface and prepare this packet for the next step which is handled by Preprocessors.

## **Preprocessors**

This component used by snort to performing some operation for the packet which is rearranged or modified before the packet goes to the detection engine cause if there is any corruption to fix.

## **Detection Engine**

It's the main and important component in this components, which the main function is that to detect the malicious activity or attack which done be perform SNORT detection rule, this rule read the data inside packet to find any matches with pre-define attack activity, if found the alert will be generated and send to next step otherwise packet dropped.

## **Logging and Alerting System**

Depending on the configuration of the snort after detection engine detect any activity, result of the detection may be either alert or log the activity or both of them, log file is used by snort to log or save any activity that detects, for this project the file stored in the **/var/log/snort** by default.

## **Output Modules**

After the alert is generated alert the output modules to save this alert, according to the way of how these users want to deal or use this data.

### **3.1.2 BASE (Basic Analysis and Security Engine)**

A web page written with PHP which provides user front-end friendly interface can be used to perform some query on the alert database.

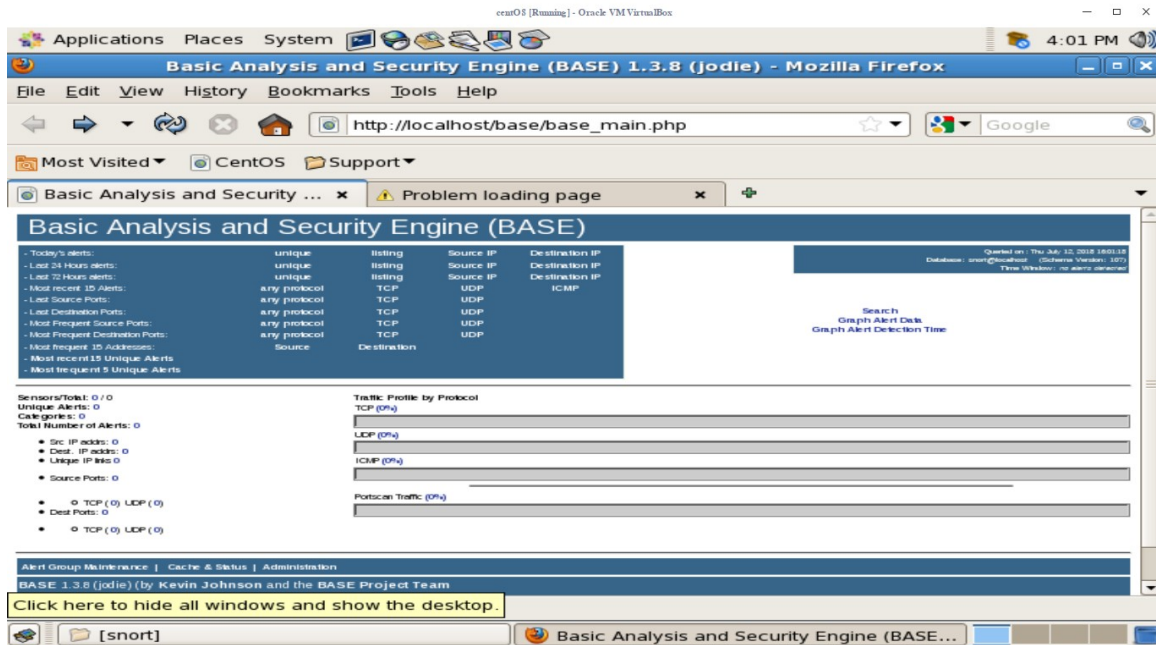


Figure 3.2: BASE interface

BASE reads both tcpdump binary log formats and Snort alert formats, BASE must be connect and configure with database to save alert which for future check, database like (MYSQL), also BASE can provide graphical overview both layer-3 and layer-4 packet information, also can provide statistical information based on IP protocol, sensor, signature, protocol. The BASE is allowed to administrator easy management the alert data, categorize the alert and delete them.

### 3.1.3 ADODB

Active Data Object Database It is a database abstraction library for PHP and Python-like Microsoft ActiveX data.

### 3.1.4 PHP

The scripting language used for web development with working with HTML, which supports use with database MYSQL.

### 3.1.5 Virtual Box

Virtual machine player which used to run multiple machines in

simultaneously mode in a single device. the project uses VM cause to implement this project on real it requires many devices so because of the lack of possibilities and hasn't hardware requirements to do it, so it's implemented in a simulation and find the best tool of simulations we have virtual machine workstation (Virtual Machine).

### **3.2 Improving the current IDSs systems**

Many ideas have been proposed for improving the current systems in use. Sekar et al. propose that a new intrusion detection rules language be developed to make creating rules for different IDSs and published in this paper “A High-Performance Network Intrusion Detection System” [23], another proposed improving proposed by Barruffi, Milano, and Montanari which by automatic responses action can be added into IDSs system to respond to attacks without relying on the administrator and allow the system to manage intrusion recovery <sup>[24]</sup>, Shen et al. proposed “a hybrid of distributed, redundant, and cross-corroborating techniques <sup>[25]</sup>. Others believe that a new system of communication protocols or a redesign of routing protocols should be developed to help combat many problems stemming from an inability to effectively trace attackers.

### **3.3 Proposed New System Architecture**

The below figure illustrates the structure of the system which based on the snort component figure.

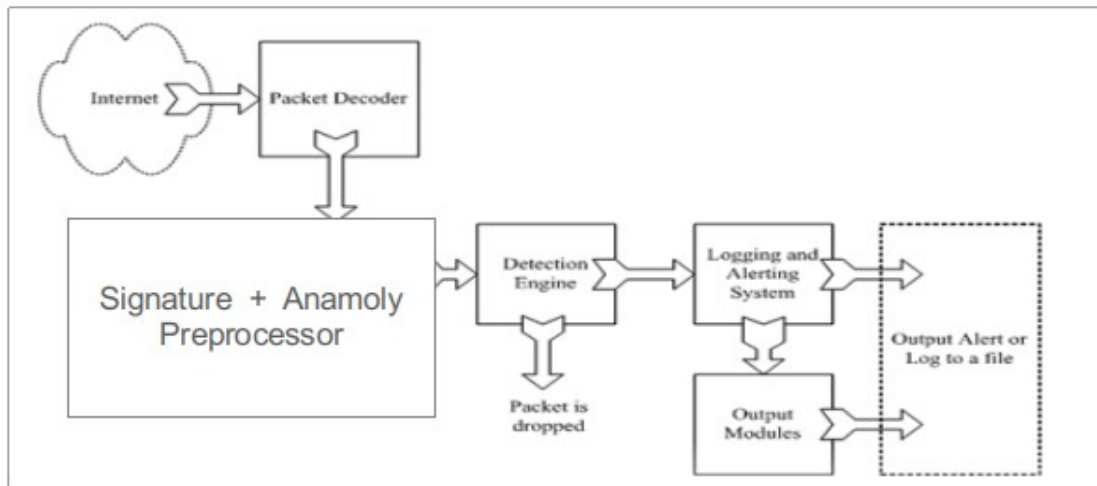


Figure 3.3: Proposed New System Architecture

The basic idea behind the new system is that by adding new preprocessor (SPADE) to the snort allow snorting to be capable to detect the anomaly behaviour on the system.

### 3.4 Statistical Packet Anomaly Detection Engine (SPADE) Preprocessor

Spade is pre-processor plug-in for the Snort intrusion detection engine, it can detect the abnormal activity in the network traffic, a spade was created by Stuart Stanford and James Hoagland, were working in the Defense Advanced Research Projects Agency (DARPA).

#### 3.4.1 The concept of Anomaly detection with SPADE

Each network have normal traffic for example if you have web Server running web service, in normal you will receive a large number of outside IP address for the single port TCP/80, another example if your running DNS service also you expect to receive a number of internal request to port UDP/53. All these activities and requested can be categorized as normal network behaviour, the “Abnormal activity” could be like getting a request from the external network to your DNS

server to port 80 TCP.

Snort signature-based detection cannot detect these activities simply cause this activity not defined in the signature rules file, SPADE preprocessor can do this thing, which allows to you to detect this activity that no prewritten signatures for, detect the zero-day attack.

### **3.4.2 Function of SPADE**

To detect abnormal activity SPADE store information about the occurrence of a different type of packet over the network, the higher value assigned to recent occurrence gradually phases out older occurrences.

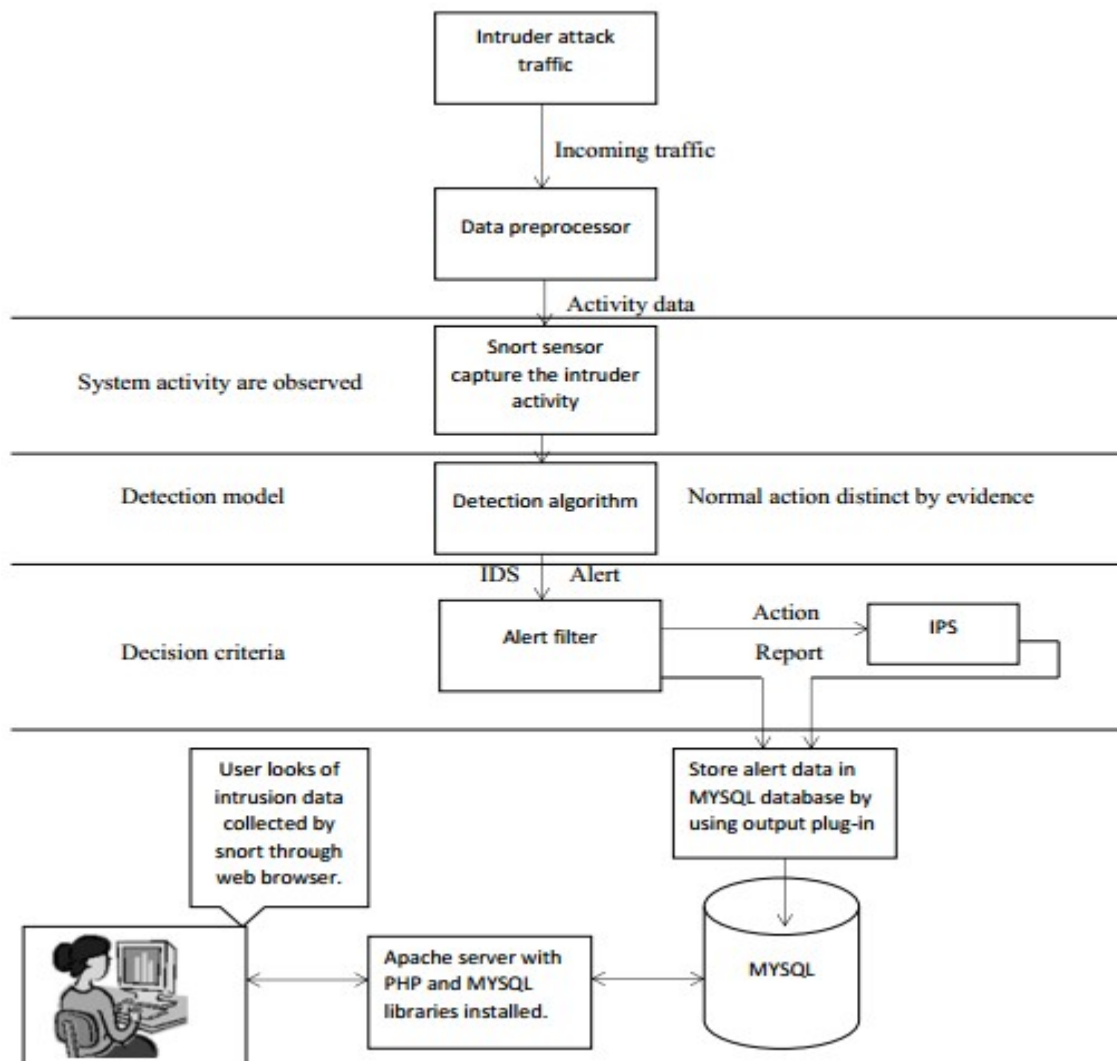
Form above example the probability table probability can provide information of a packet to the DNS server on port 53 is 10% ( $P(\text{dest\_ip}=\text{DNS\_SRV}, \text{dest\_port}=53) = 10\%$ ) whereas the probability of a packet to the DNS server on port 80 is 0.1% ( $P(\text{dest\_ip}=\text{DNS\_SRV}, \text{dest\_port}=80) = 0.1\%$ ). by using and compare this information can detect something uncommon about packets, SPADE converts this probability into “raw anomaly score” and then into a “relative anomaly score”, this production number which is 0 or 1, which can be easy to the comparison.

The calculation will be the probability of packet X the “raw anomaly score”  $a(X)$  is equal to  $-\log_2(P(X))$  to find “relative anomaly score” the divide with the maximum possible of “raw anomaly score”, this produces two results 0 or 1, zero is normal while one not normal activity.

### **3.5 System Process View**

The below figure show clearly how actually IDS system work, after

install and configure the necessary tools and software packages.



**Figure 3.4: System process view**

The figure above demonstrates the IDS processes since the traffic passed through the network interface till this traffic classified, checked and reported by IDS, the steps as follow:

The attacker packet with a specific signature pass through the network interface where snort sensor is deployed and configure, then the packet goes through data pre-processor, the output of this is activity data that reside in the packet will by mining it will send to the snort sensor to



capture intruder activity , this activity passed in the detection algorithm the detection algorithm distinct Normal action against the abnormal activity if the misuse or abnormal activity detect the IDS will generate alert the alert with three option, one is to take action against this activity which blocks it and in this case this is IPS (Intrusion Prevention System) if IPS implemented, second is to report this activity without action will take, finally may take both actions together.

### 3.6 Evaluating IDS

After deploying and implement the IDS it a test and evaluate to make sure the system is running probably and with full functionality, to carry out this we can use many tools and data, and it's discussed below.

#### 3.6.1 Using CommView to Generate Packet

It's used to generate customized network packet for specific purpose to test IDS, in this test we create the TCP/UDP/ICMP packet which this packet deviate from normal packet behaviour and it sends over the network, this packet must be capture by IDS according to rule the is wrote beforehand and then generate alert.

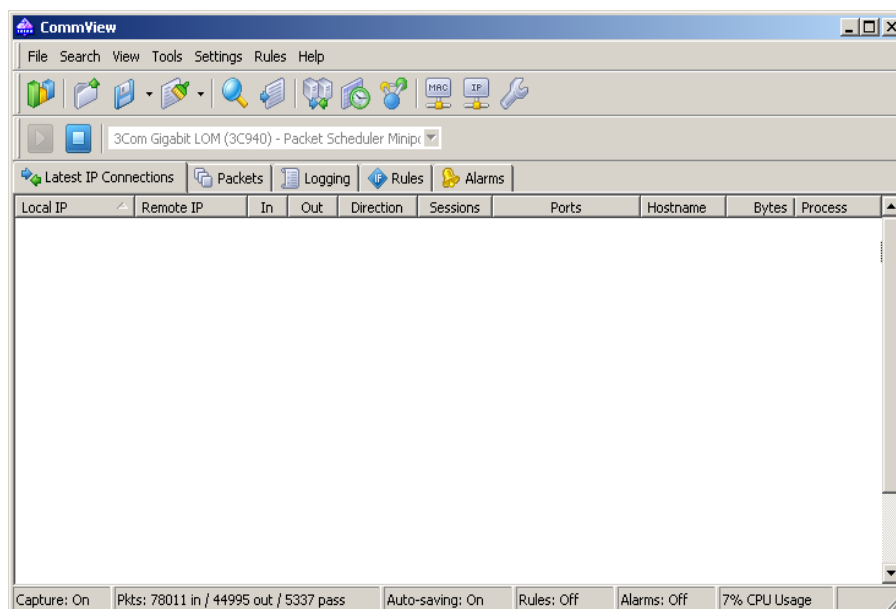


Figure 3.5: CommView – Main page

### **3.6.2 DARPA Intrusion Detection System Data Set**

Lincoln Labs' intrusion detection evaluation is another way that used for testing and Evaluation the SNORT IDS system, Lincoln offered dataset for testing known by DAPRA dataset, testing IDSs by using network traffic and audit logs collected in simulation network, the IDS test with this data to find out the number of alert that will be generate <sup>[26]</sup> .

There is two type of intrusion DARPA Intrusion Detection Evaluation: an off-line evaluation and a real-time evaluation.

Intrusion Detection System was testing on the offline mode the test by using network traffic and audit logs collected on a simulation network, the system process this data and identify the intrusion attack activity the in a file usually the file is (tcpdump) file.

Additional, type of testing in real-time evaluation is sent to AFRL (Air Force Research Laboratory) which IDSs system inserted into network tested to detect any attacks attempts.

### **3.7 Validation and Sureness**

Testing the system using DARPA data set will be Performing to IDS system before and after the Pre-processor is added and compare the result of the total alert the generated by, also to make sure that IDS systems are working probably.

### **3.8 Result presentation**

At the end of the above system processes, it should to presenting the results in order to compare them against objectives of research that must be achieved.

### **3.9 Summary**

This chapter illustrates up to the research methodology which includes a detailed description instrumentation (tools and the system environment) that used to implement the proposed system, also provide the new system process overflow and how to evaluate and validate the new system.

# **Chapter Four**

## **RESULTS AND DISCUSSIONS**

## CHAPTER FOUR

### Results and Discussions

#### Introduction

Here in this chapter include the discussion and the result of testing After install and configure needed package and software, for carrying out this work, it has been parted in section first implement IDS with signature-based detection, second implement the anomaly detection and finally implement and test the IDS with this two techniques together.

#### 4.1 Operate With SNORT

Snort can be operated in two basic modes: packet sniffer mode and NIDS mode. The packet sniffer mode is like dumping or TCP dump, when work with the sniffer mode snort will sniff all packet that passed through the network, also can log all this sniffed packet in file which can be view or check later, snort that operates in this mode no intrusion detection activity is done, the intrusion detection is done when SNORT operate NIDS mode, which snorts use rule for detection the intruder activity on the network.

##### 4.1.1 Sniffer Mode

Simply this mode performed like normal tcpdump program, SNORT capture and display any packet that passed through the network, this packet can be displayed in a different level of detail on the console.

To issue this mode you can use the following command.

```
[root@Server ~]# snort -v
```

```

root@Server:~
File Edit View Terminal Tabs Help
[root@Server ~]# snort -v
Running in packet dump mode

--== Initializing Snort ==--
Initializing Output Plugins!
***
*** interface device lookup found: eth0
***
Initializing Network Interface eth0
Decoding Ethernet on interface eth0

--== Initialization Complete ==--

    ,,_
   o" )~
  ,',',
 t-team

    -*> Snort! <*-
    Version 2.8.6.1 (Build 39)
    By Martin Roesch & The Snort Team: http://www.snort.org/snort/snort-team
    Copyright (C) 1998-2010 Sourcefire, Inc., et al.
    Using PCRE version: 6.6 06-Feb-2006

Not Using PCAP_FRAMES

```

Figure 4.1: Snort Sniffer Mode

```

root@Server:~
File Edit View Terminal Tabs Help
=====
07/12-17:17:52.449083 10.0.0.1 -> 10.0.0.2
ICMP TTL:64 TOS:0x0 ID:59328 IpLen:20 DgmLen:60
Type:0 Code:0 ID:1 Seq:8 ECHO REPLY
=====
07/12-17:17:53.448224 10.0.0.2 -> 10.0.0.1
ICMP TTL:128 TOS:0x0 ID:762 IpLen:20 DgmLen:60
Type:8 Code:0 ID:1 Seq:9 ECHO
=====
07/12-17:17:53.448335 10.0.0.1 -> 10.0.0.2
ICMP TTL:64 TOS:0x0 ID:59329 IpLen:20 DgmLen:60
Type:0 Code:0 ID:1 Seq:9 ECHO REPLY
=====
07/12-17:17:54.448122 10.0.0.2 -> 10.0.0.1
ICMP TTL:128 TOS:0x0 ID:763 IpLen:20 DgmLen:60
Type:8 Code:0 ID:1 Seq:10 ECHO
=====
07/12-17:17:54.448149 10.0.0.1 -> 10.0.0.2
ICMP TTL:64 TOS:0x0 ID:59330 IpLen:20 DgmLen:60
Type:0 Code:0 ID:1 Seq:10 ECHO REPLY
=====

```

Figure 4.2: Sniffed Packet Detail

After sniffing done (Stop sniff process) SNORT provides detail about all sniffed packet such as:

- Date and time packet was captured
- Source and Destination IP address
- Source and Destination of packet Port number

- Packet ID
- Received Packet
- Analyzed packet
- Dropped packet

And other information that related with the packet, also Sniffer mode can be used to capture packet in a different level of detail such as information about application layer data and information about protocol TCP, UDP and ICMP.

```
>> snort -dv
```

This command for display packet with data link layer

```
>> snort -dev
```

Will display all the packet in hexadecimal format and ASCII format.

And many other command and option can be used in different purpose check out this book for addition command [<sup>12</sup>].

#### **4.1.2 Network Intrusion Detection Mode**

In the intrusion detection system the all the packet that passed through the network it checked and applied to rules, these rules contain some signature of the attack, with this signature if the packet is matched it the alert will generate, otherwise if not match any rule the packet drop and not logged in the system.

The Snort get the rule and configuration from configuration file it's saved as snort.conf, this file contains all rule path the save in rules directory.

To activate SNORT in network intrusion detection mode you can issue

the following command:

```
[root@Server ~]# snort -c /etc/snort/snort.conf -i eth0
```

When this command is running it reads all settings and configuration such as rules, preprocessor and network interface settings.

#### **4.1.3 Working with rules**

All Snort operations are based on the rule of detection intruder, this rule is known as signature rules, it divides into two main parts: rule header and rule option.

According to the need, it can modify the rules, the rule header contains the specific action that will be taken if the packet matches the rule, the option contains an alert message and which part of the package generates the alert message.

Header of rule contains (Action, Protocol, Source Address, Source Port, Direction, Destination Address, and Destination Port).

The action contains the action that should be taken for the matched packet, usually alert or log, with the below rule will clearly show the components

```
Alert ip any any -> any any (msg : "IP Packet Detected";)
```

Action = Alert, Protocol = IP, Source Address = any address, Source Port = any port, Direction = -> , Destination Address = any port , Destination Port = any port .

#### **4.2 Implementation Part 1: Experimentation Signature Rule Set**

Signature based detection snort looks for attack signatures and matches this signature with predefined rules stored in a database or (rules) file, which will be later required in snort configuration file for detection.



Seven snort signature will be generated and test by snort IDS system, the uses of packet generator are to create and generate packet according to a specific parameter, the common value for some parameter are as follow:

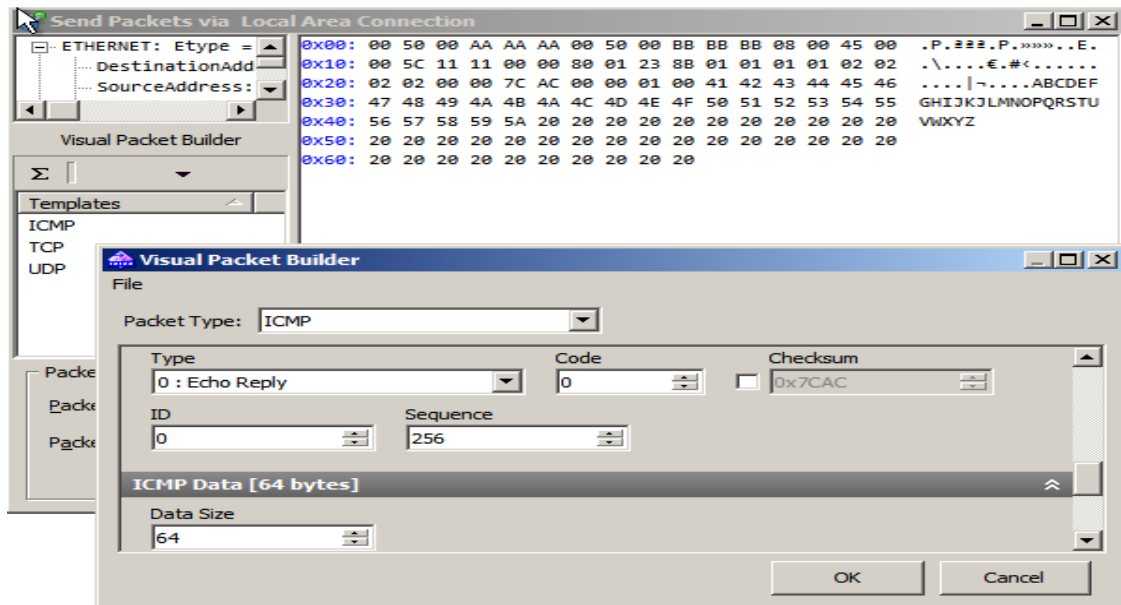
- Internet Protocol version = 4
- Header length of IP = 20 bytes
- Service Type = 0
- Source IP address = 10.0.0.2
- Destination IP address = 10.0.0.1

#### **4.2.1 Signature 1 : DDOS (TFN) ICMP possible communication**

This signature creates an event when ICMP traffic sent between TFN (Tribe Flood Network) when this flood of a packet sent to the network only explanation for this event is that there is an attempt for denial of service attack (DOS) attack.

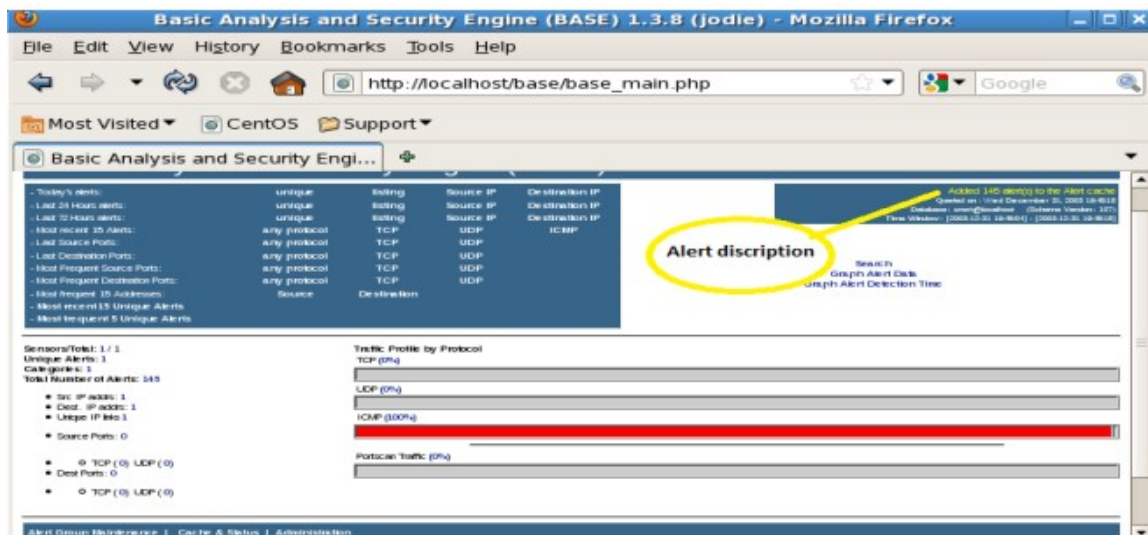
To generate this packet I can be used this parameter (ID=0, Type=0, packet-size=105) as shown in figure 12.

The object from this is Attempts to "flood" a network, thereby preventing legitimate network traffic and attempts to prevent a particular individual from accessing a service.



**Figure 4.3: DDOS (TFN) ICMP possible communication**

When the packet sent through the network and detected with snort system the BASE provides the following description about the detected packet.



**Figure 4.4: BASE alert of ICMP traffic**

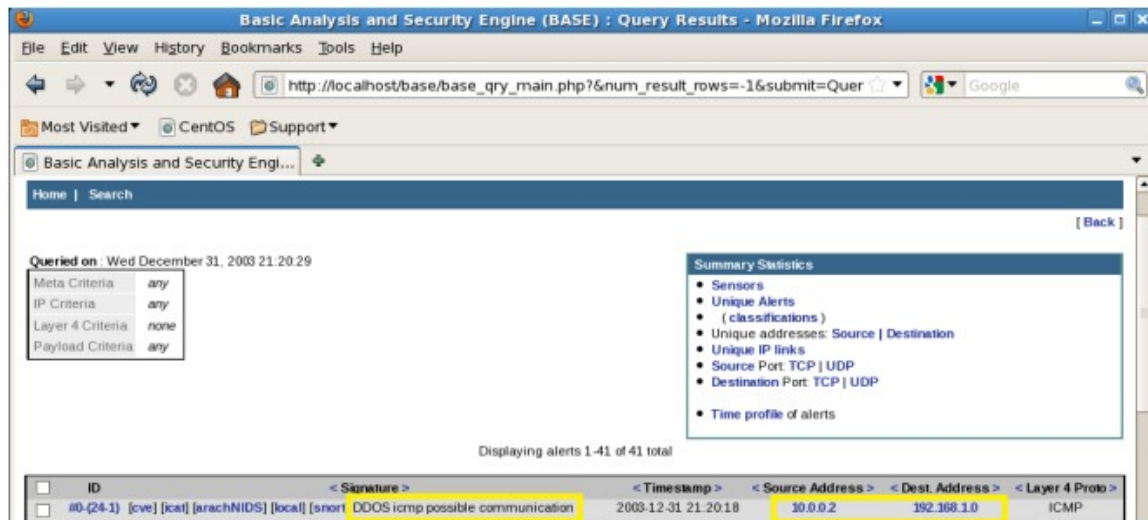


Figure 4.5: BASE Alert (DDOS ICMP possible communication)

#### 4.2.2 Signature 2: MISC source port 53 to 1024

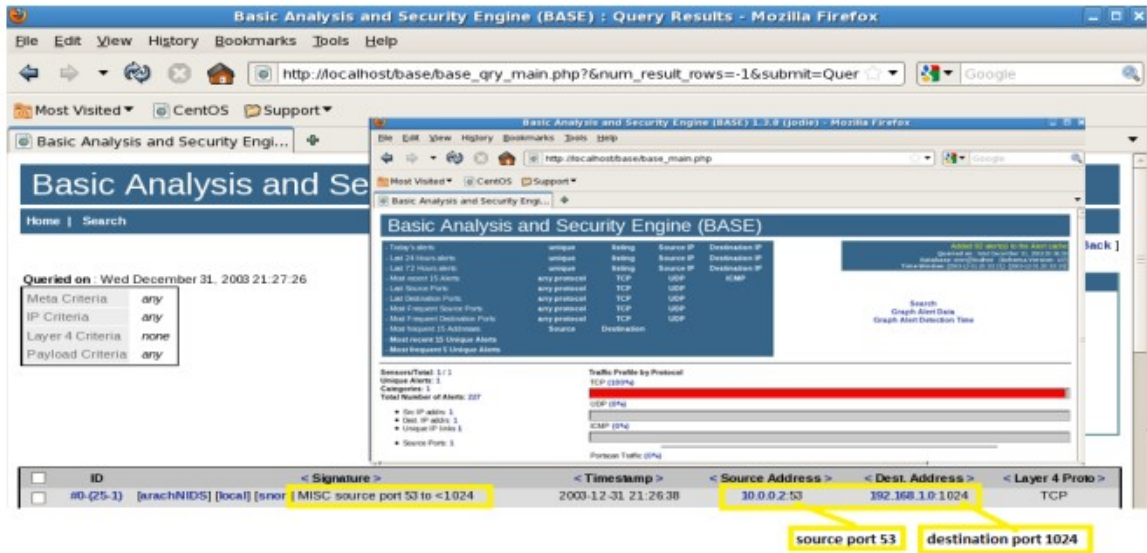
The event of detection of this attack it will be generated when the non-legitimate traffic passed through the network through the firewall.

Normally the DNS uses the UDP protocol, DNS traffic from port 53 using either UDP or TCP should be to a port above 1023. An attacker could use a source port of 53 for TCP connections to bypass a poorly configured firewall.

One explanation of such packets being transmitted in the network could be an attempt to 'spoofing DNS attack'

To generate the packet the parameters are set as follow:

- TCP packet with source port 53 to 1024 with SYN flag set
- Header length with 5 and no checksum option.

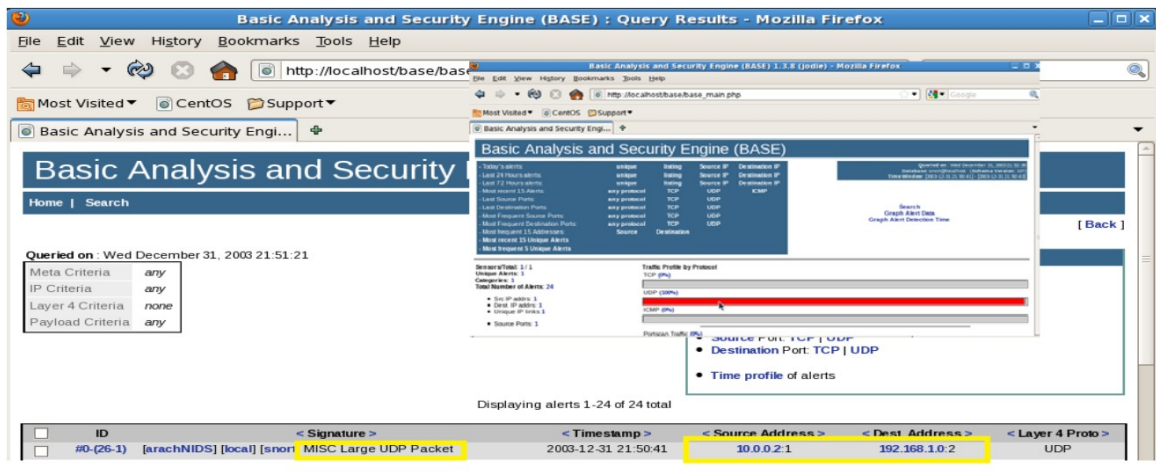


**Figure 4.6: BASE Alert (DDOS ICMP possible communication)**

### 4.2.3 Signature 3: large UDP Packet

Snort generate an alert when large UDP packet detected in a normal scenario the UDP packet payloads small than 4000 since this protocol designed to be used for the transmission of smaller payloads, the probability for payloads more than 4000 that means is the possibility of ‘DOS – Denial of Service’.

To generate this packet it can set the packet data size more than 4000 just.



**Figure 4.7: BASE Alert (large UDP Packet)**

#### 4.2.4 Signature 4 and 5: BAD TRAFFIC TCP, UDP port 0 traffic

IDS (e.g. Snort system) generate alert when receiving TCP, UDP packet with destination filed port (0), at all normal situation the TCP, UDP packet doesn't have destination port (0), the explanation of this activity is which can be one of the attack phases is that is (reconnaissance activity) it's used by attacker to indicate the existence of host in the network at a particular address which is listening to requests as an as a prelude to an attack.

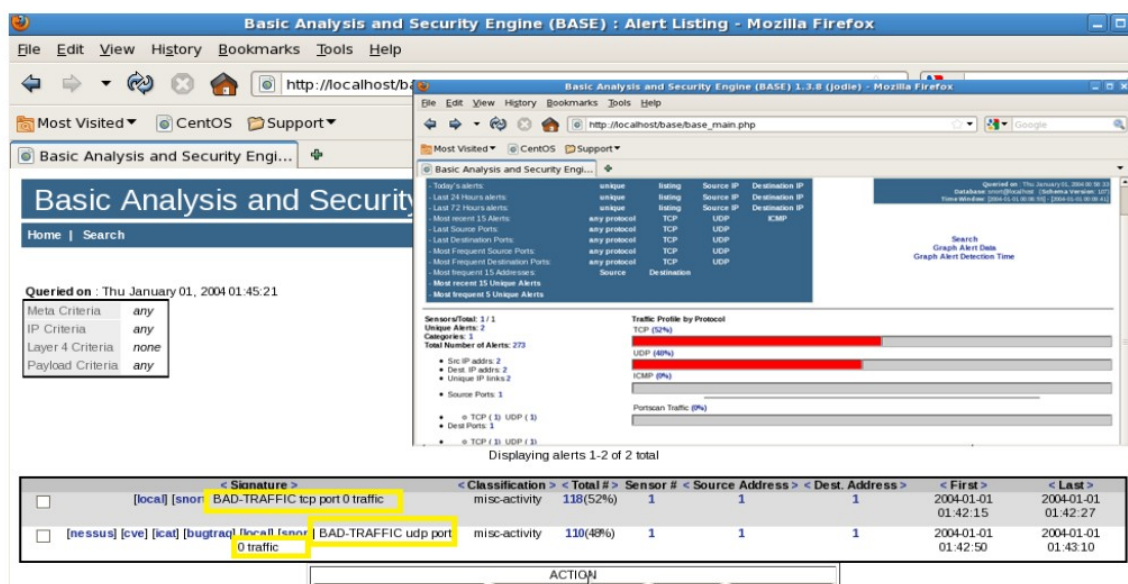


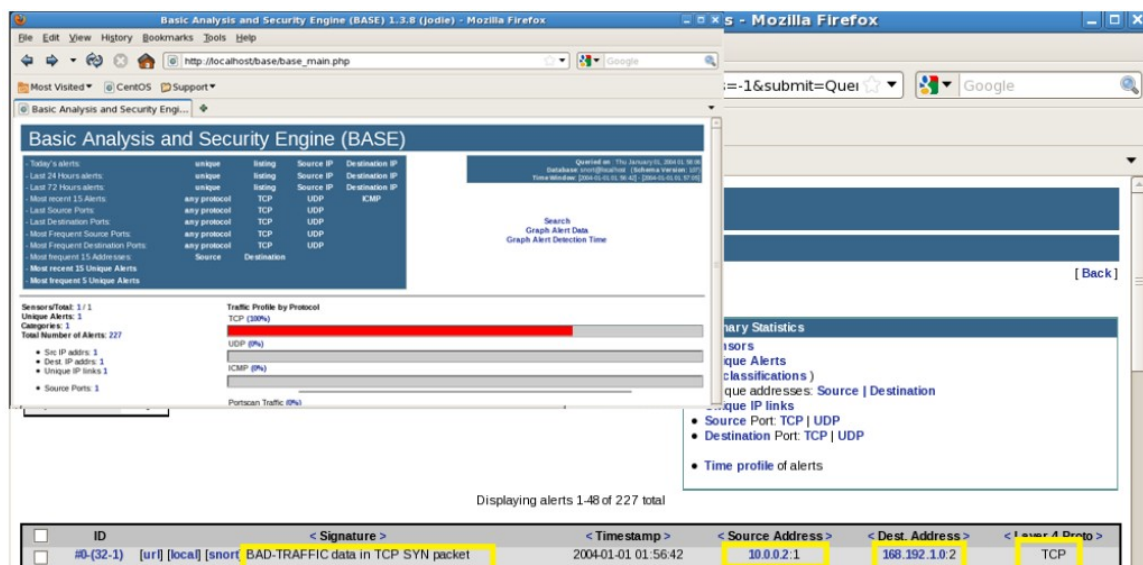
Figure 4.8: BASE Alert (BAD TRAFFIC TCP, UDP port 0 traffic)

#### 4.2.5 Signature 5: BAD TRAFFIC data in TCP SYN packet

When TCP packet contain SYN flag is dedicate that the source host want to request for a connection establishment with destination host, in normal scenario this packet doesn't contain any data, these packet used for synchronizing sequence numbers in communication time, SYN packet size is 6 byte if the size of packet increased this may indicate that there attempt of denial of service attack.

To generate this packet use TCP packet with SYN flat set and data size

greater than 6 bytes.



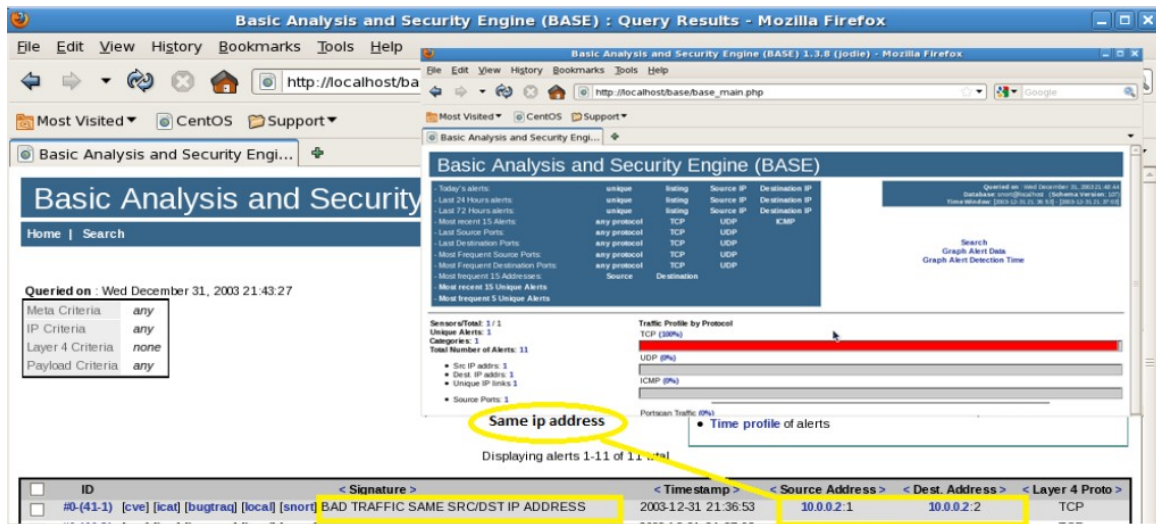
**Figure 4.9: BASE Alert (BAD TRAFFIC data in TCP SYN packet)**

#### 4.2.6 Signature 6: BAD TRAFFIC same SRC/DST IP address

IDS system like snort generate an alert when the packet passed through snort with the same source and destination IP address, normally TCP packet doesn't have the same IP address for source and destination host except loopback packet.

Some of the TCP stacks crashed with TCP SYN flag packet that with the same Source and Destination IP when the TCP stack is crash the attacker can send a packet with spoofed IP address, which is the IP of the Destination Host.

Simply to create this packet which you can set a flag and the source and destination IP addresses.

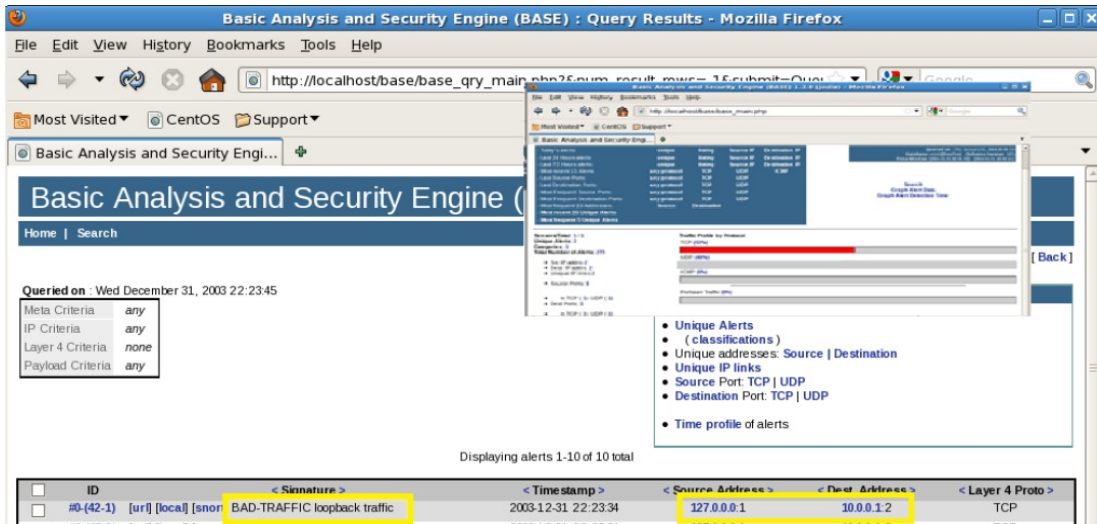


**Figure 4.10: BASE Alert (BAD TRAFFIC same SRC/DST IP address)**

#### 4.2.7 Signature 7: BAD TRAFFIC loopback traffic

Loopback packet in the normal and general scenario is destined to localhost and detected by loopback interface which is lo0. snort and other IDS generate an event when detect loopback packet that sent over the network this packet may indicate there is unauthorized network use, reconnaissance activity, the attacker use spoofed source IP address in the packet.

For generating this packet TCP packet with source IP as 127.0.0.0 and destination IP address is IP of IDS server.



**Figure 4.11: BASE Alert (BAD TRAFFIC loopback traffic)**



### 4.3 Experimentation signature base detection Evaluation

Snort signature base detection the rule can be set by two way you can customize the rule and include the rule that you need in your IDS system secondly you can use full snort rules which provide for free by snort community, this kind of rule helpful to detect 0 day attack according to predefined rules, cause most of it came with updated to the last signature of attack.

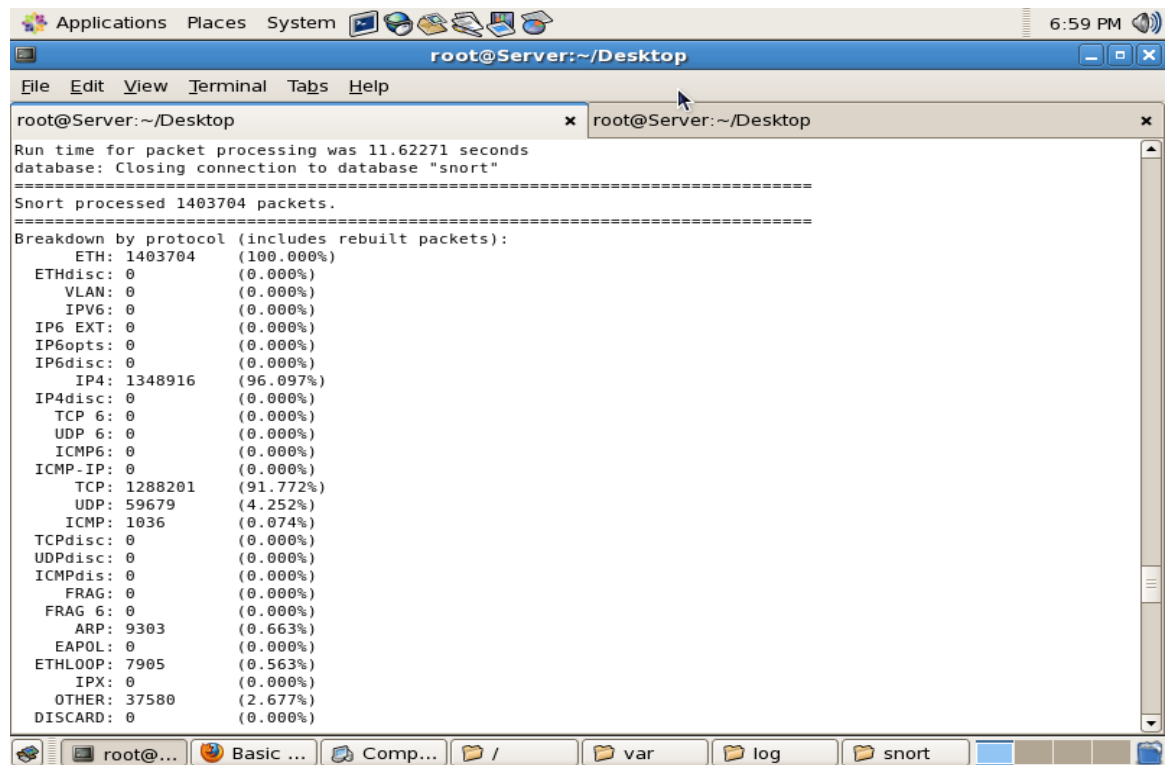
For the experiment of the signature-based detection method, all snort signature rule is imported and stored in the rule directory and were called in the snort configuration file.

In this experiment we run snort and passing DARPA 1999 dataset to snort sensor with the following command:

```
[root@Server ~]# snort -r darpa.tcpdump -c snort.conf
```

This means snort read the data from darpa dump file and apply the snort configuration that in .conf file for detection any attack activity.

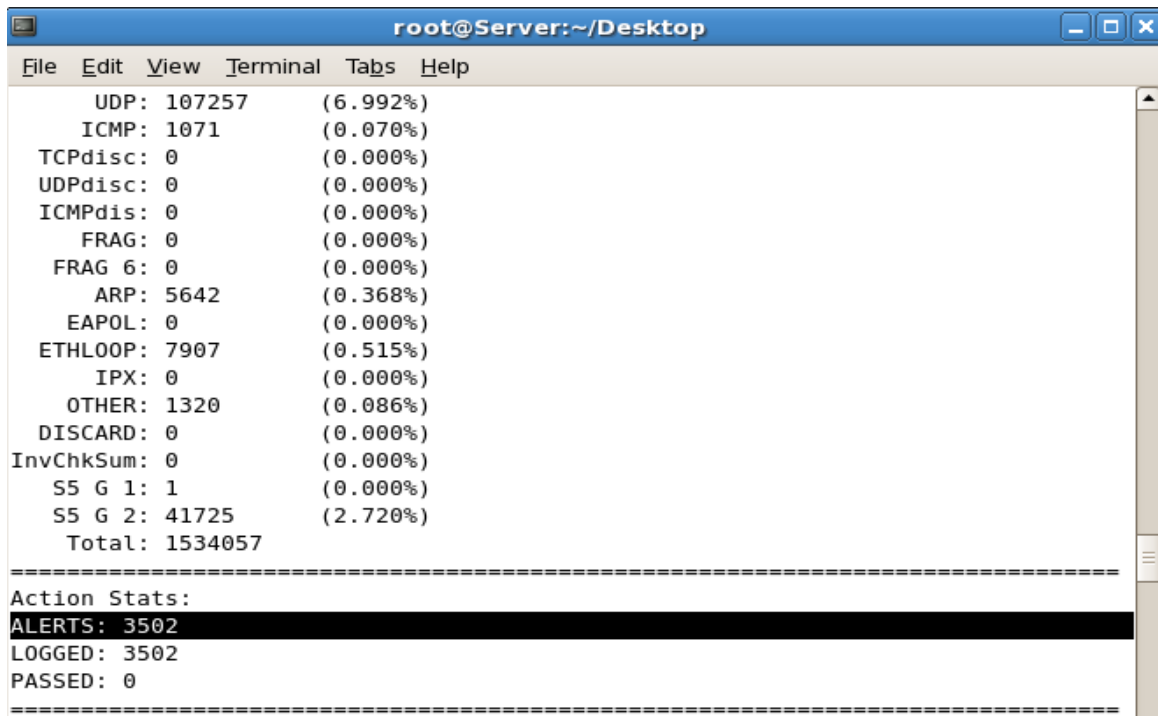
After running the command and start snort IDS system the result as shown:



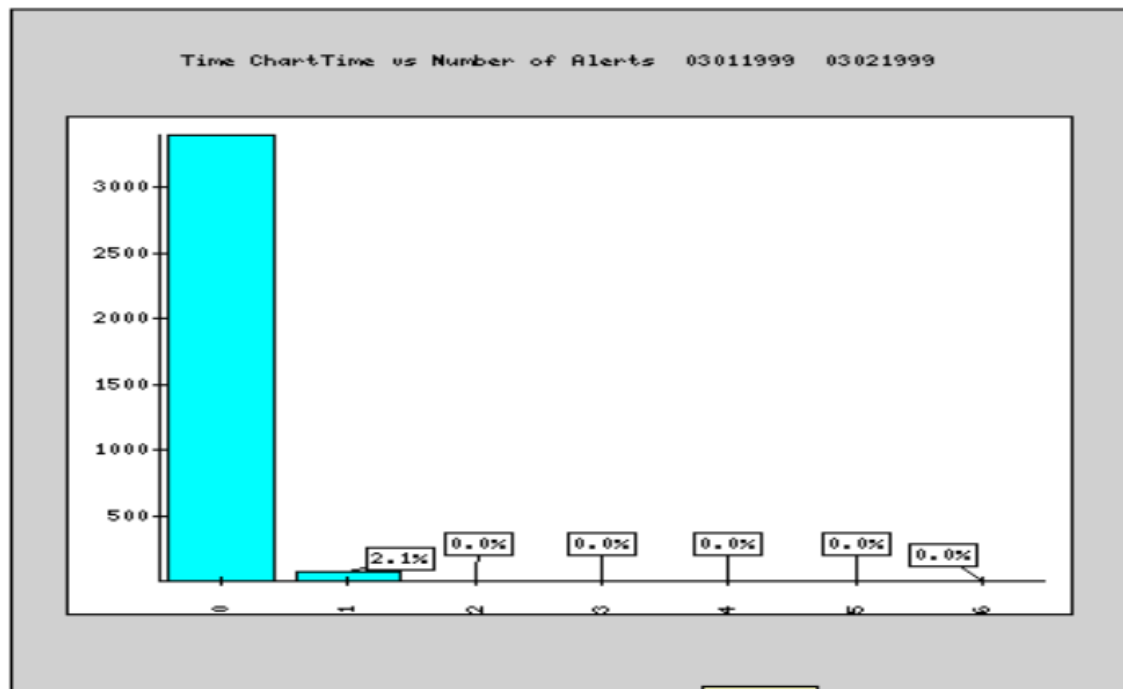
**Figure 4.12: Snort Result number of Packets Processed**

The above figure shows the total number of the packet was founded on the *DARPA tcpdump* file, these packets are processed by snort Signature pre-processor to be passed for the next stage which is detection engine which on figure 21.

The figure21 show that a total number of altering that have been detected by detection engine and how many numbers of these alert have been logged.

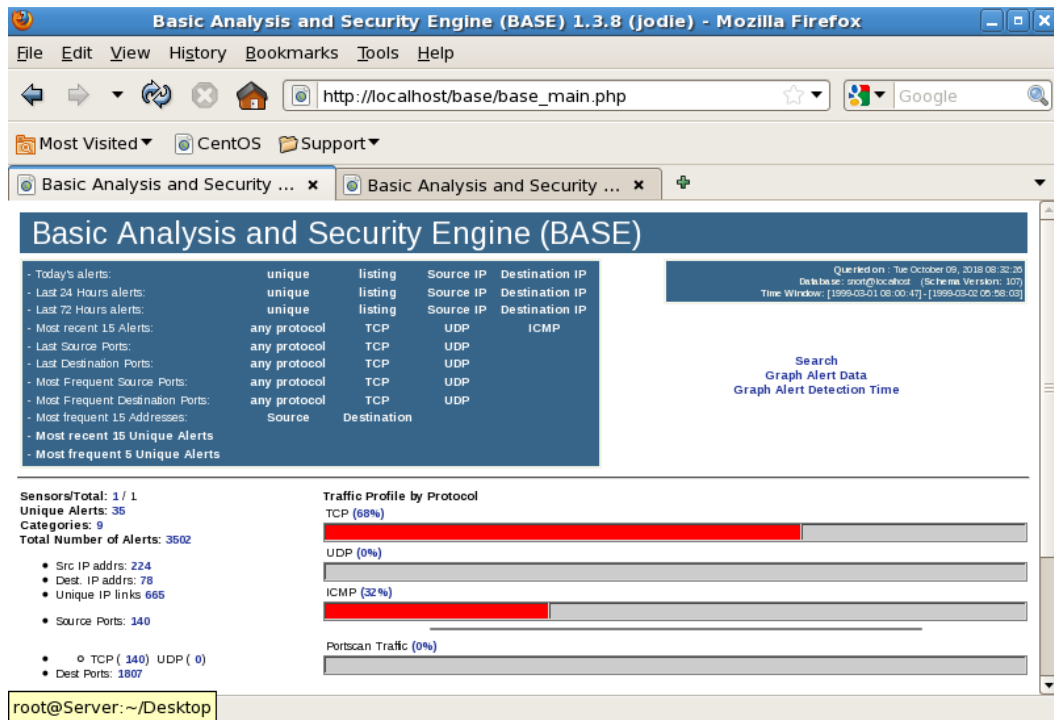


**Figure 4.13: Number of Alerts for Darpa Signature test**



**Figure 4.14: Time Chart vs Number of Alert (Signature Detection test)**

To get more viewers on the result of the IDS system we can use Basic Analysis and Security Engine (BASE) to show more detail, as shown on figure22 below.

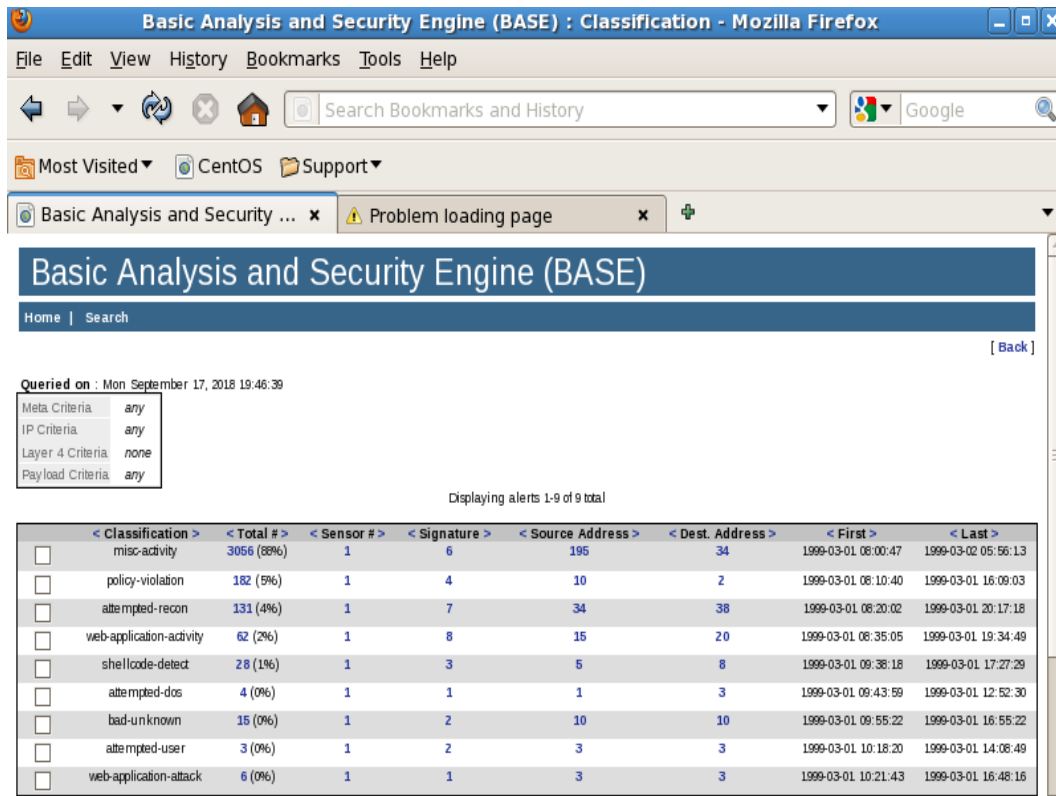


Fi

**Figure 4.15: BASE Alerts about DARPA Evaluation**

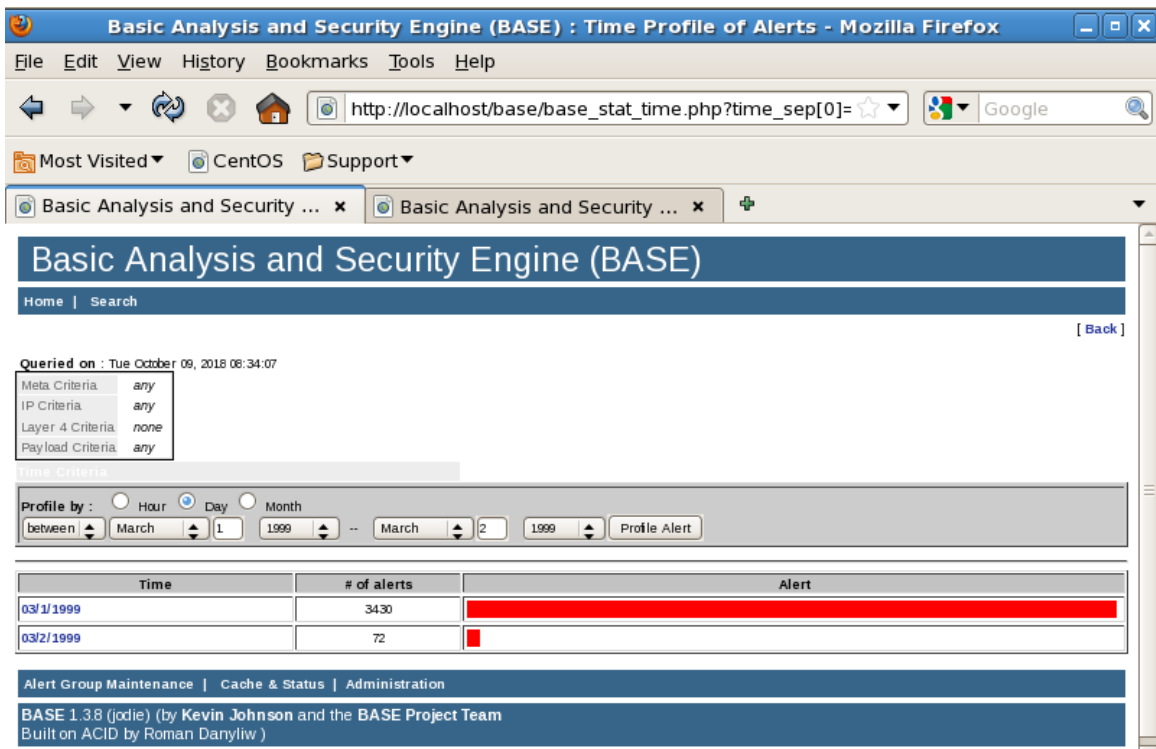
The base figure shows that the total number of alert that events created and is equal to *3502 alerts*.

Correspondingly, you can show the type and category of that attack that has been detected with IDS system which is 9 types shown in the figure below.

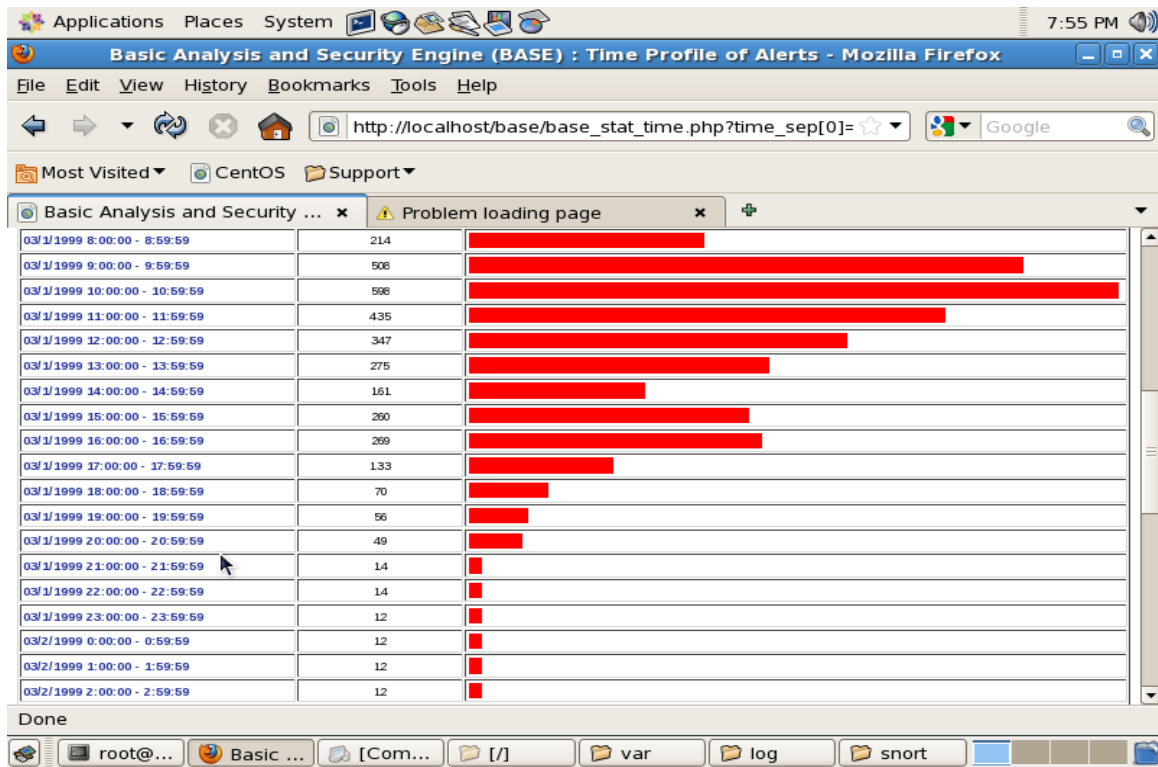


**Figure 4.16: Alert Category**

Next figure also shows the number of alerts that have been captured per day



**Figure 4.17: number of alert per-day**



**Figure 4.18: Time of alert per-hour**

## 4.4 Experimentation SPADE Anomaly base detection Evaluation

### SPADE preprocessor

The basic operation of this will be that Spade will monitor the network and report anomalous events, SPADE alert will be configured to generate specify event alert, for example, packet (X) over a certain value will generate an alert, sending a standard Snort alert through the normal Snort alerting mechanisms.

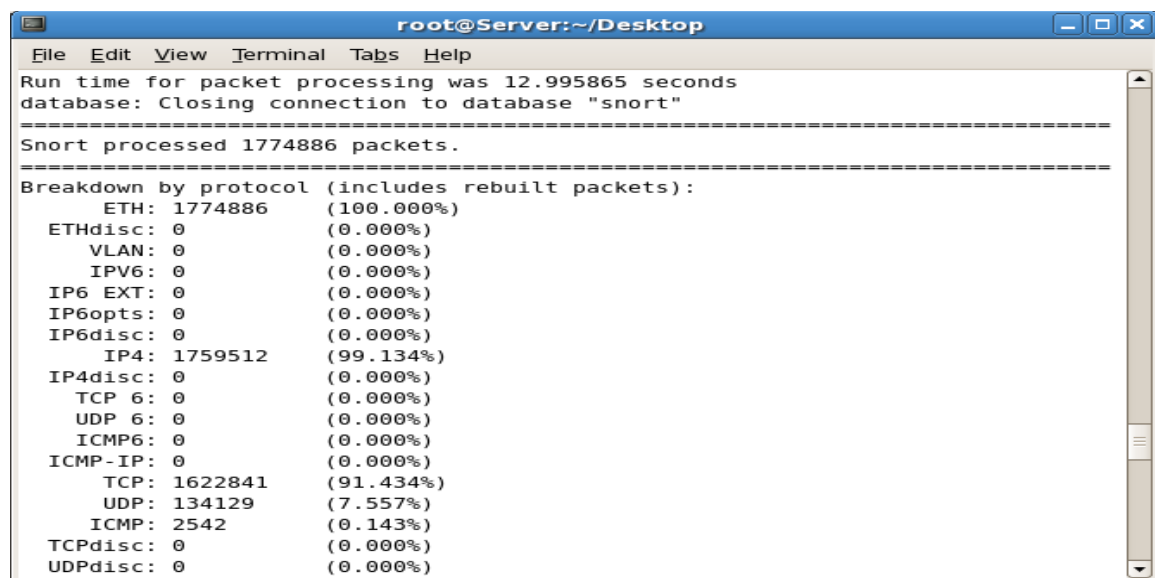
In this work SPADE will be tested by using also DARPA data-set which contain network packet within TCP dump file, this file contains an anomaly packet which is most of it is an ICMP misuse packet, which can be used by an attacker to perform Denial of Service attack.

After adding and plug the SPADE pre-processor, run snort and passing DARPA 1999 dataset to snort sensor with the following command:

```
[root@Server ~]# snort -r darpa.tcpdump -c snort.conf
```

This means snort read the data from DARPA dump file and apply the snort configuration that in .conf file for detection any attack activity.

After running the command and start snort IDS system the result as shown:



```
root@Server:~/Desktop
File Edit View Terminal Tabs Help
Run time for packet processing was 12.995865 seconds
database: Closing connection to database "snort"
=====
Snort processed 1774886 packets.
=====
Breakdown by protocol (includes rebuilt packets):
  ETH: 1774886      (100.000%)
  ETHdisc: 0       (0.000%)
  VLAN: 0          (0.000%)
  IPV6: 0          (0.000%)
  IP6 EXT: 0       (0.000%)
  IP6opts: 0       (0.000%)
  IP6disc: 0       (0.000%)
  IP4: 1759512     (99.134%)
  IP4disc: 0       (0.000%)
  TCP 6: 0         (0.000%)
  UDP 6: 0         (0.000%)
  ICMP6: 0        (0.000%)
  ICMP-IP: 0      (0.000%)
  TCP: 1622841    (91.434%)
  UDP: 134129     (7.557%)
  ICMP: 2542      (0.143%)
  TCPdisc: 0      (0.000%)
  UDPdisc: 0      (0.000%)
```

**Figure 4.19: Result of Snort with SPADE number of Packets Processed**

The above figure shows the total number of the packet was founded on the *DARPA tcpdump* file, these packets are processed by snort Signature and SPADE pre-processor to be passed for the next stage which is detection engine which on figure 28.

The figure28 show that a total number of altering that have been detected by detection engine and how many numbers of these alert have been logged.

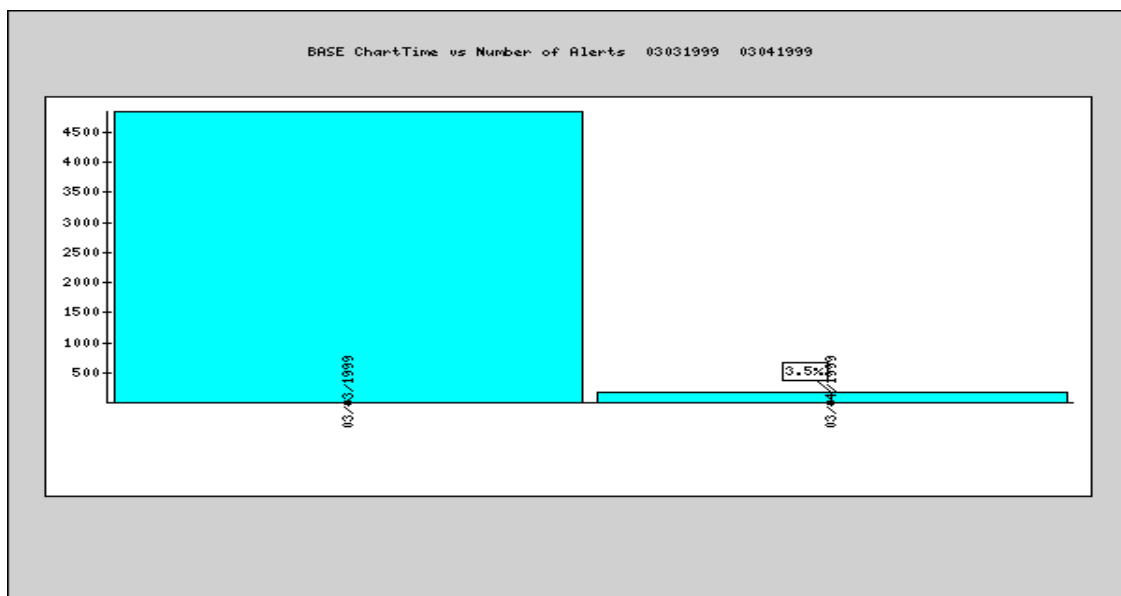
```

root@Server:~/Desktop
File Edit View Terminal Tabs Help
Total: 1774886
=====
Action Stats:
ALERTS: 5054
LOGGED: 5054
PASSED: 0
=====
Frag3 statistics:
  Total Fragments: 0
  Frags Reassembled: 0
  Discards: 0
  Memory Faults: 0
  Timeouts: 0
  Overlaps: 0
  Anomalies: 0
  Alerts: 0
  Drops: 0
  FragTrackers Added: 0
  FragTrackers Dumped: 0
  FragTrackers Auto Freed: 0
  Frag Nodes Inserted: 0
  Frag Nodes Deleted: 0
=====
Stream5 statistics:

```

**Figure 4.20: Number of Alerts**

For above graph as seen the number of alerts is increased against the alert that generates by snort signature pre-processor on the first experiment, the increase is a result of a number of packets that detected with signature pre-processor and SPADE pre-processor, more details on next figures.



**Figure 4.21: Time Chart vs Number of Alert per-day**



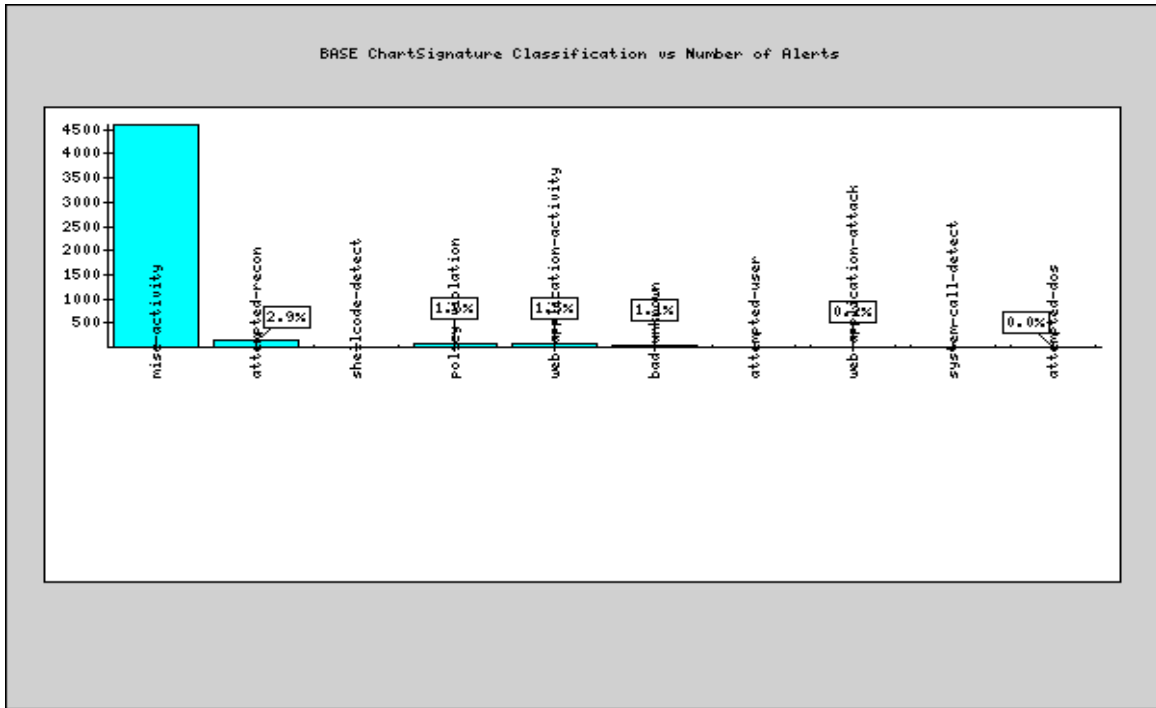


Figure 4.22: Alert Categorization vs Number of Alerts

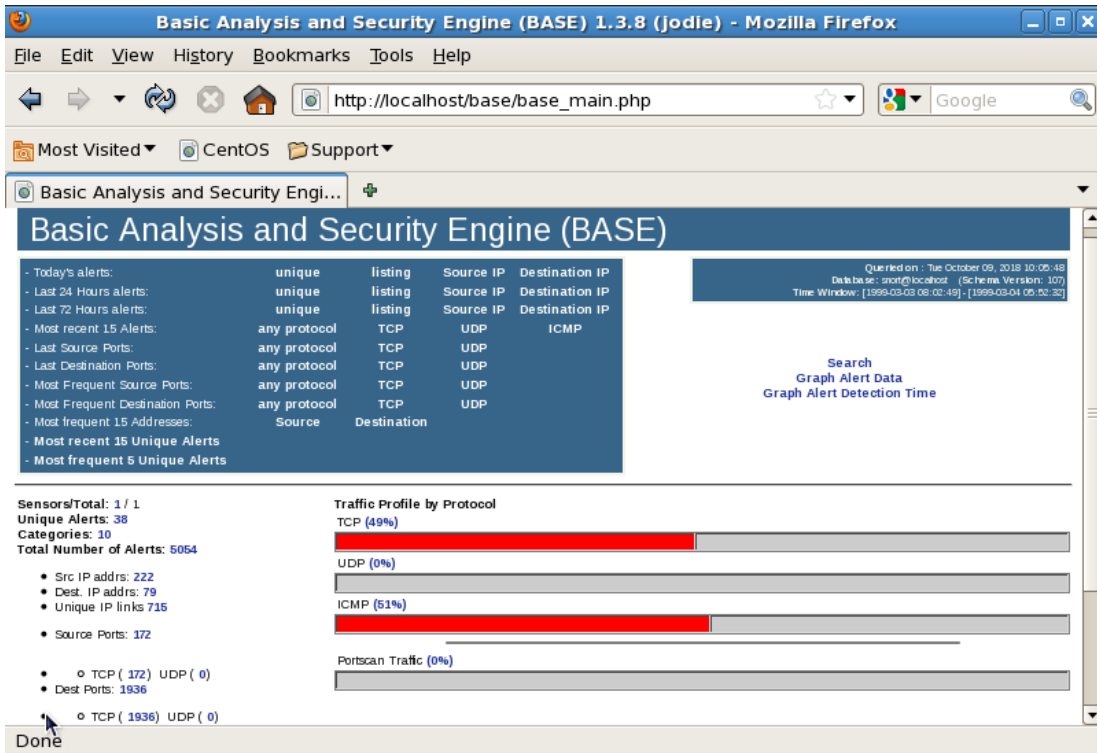
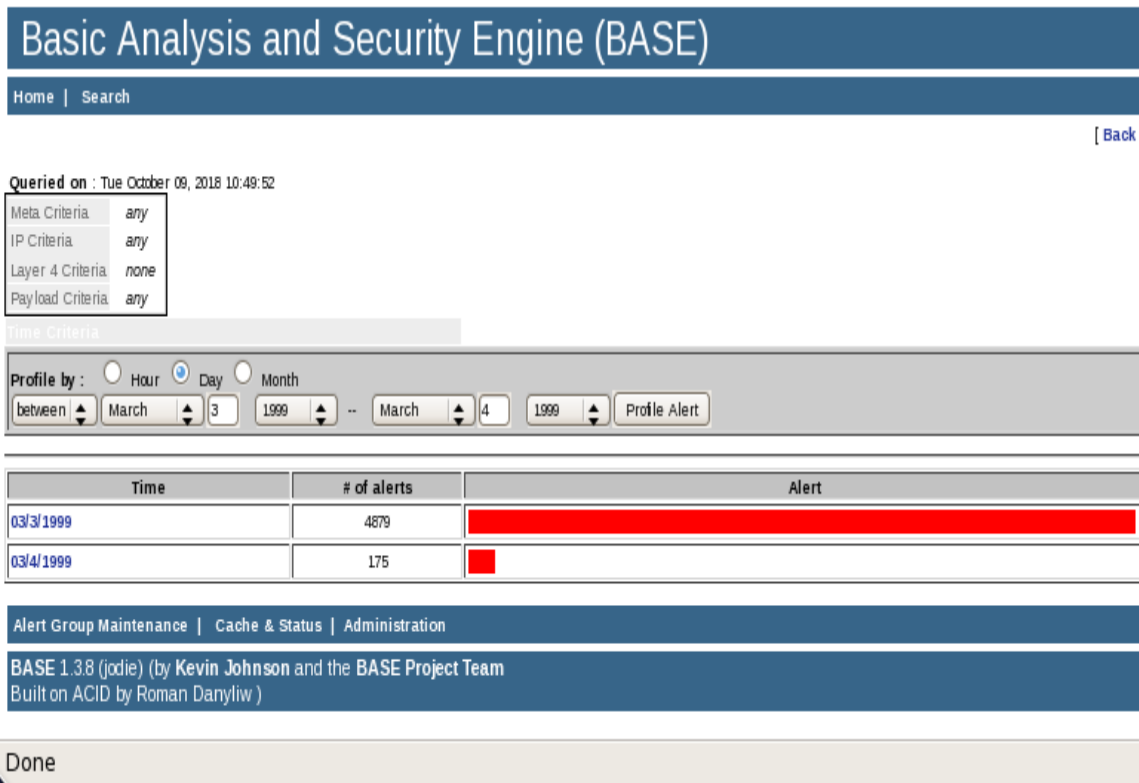
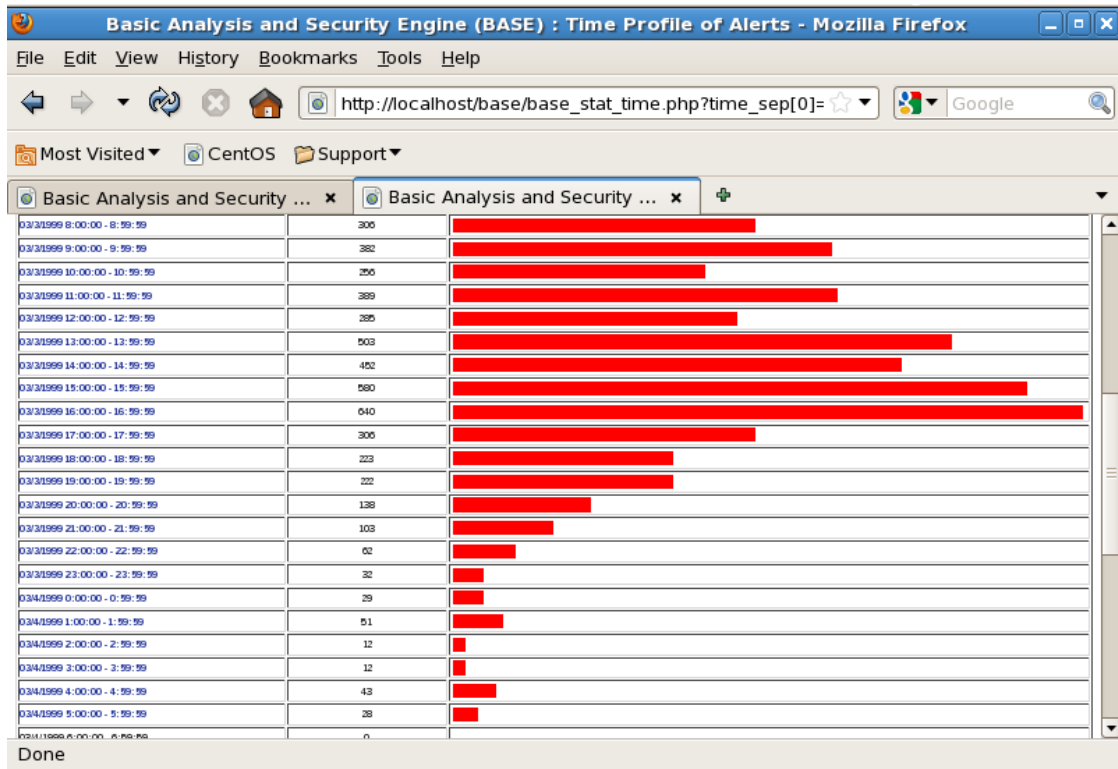


Figure 4.23: BASE Alerts about DARPA Evaluation on anomaly pre-processor

In this time the alerts show that the number of unique alerts was increased from a previous time and the ICMP detection percentage is increased to 51% this explains that there is an abnormal activity on passed packets.



**Figure 4.24: number of alert per-day**



**Figure4.25: Time of alert per-hour**

## 5. Result and Discussion

A network intrusion detection system like SNORT has typically used signature detection, which by matching the pattern in the network traffic with a predefined pattern of attack, but the method have dis-advantage whereby cannot detect the novel attack. An alternative approach is anomaly detection method which based on normal traffic and signals any deviation from this model as suspicious, the process of combine these two techniques can offer many advantages like increase the detection rate, this mission can be done by add preprocessor to the SNORT architecture.

After configure the snort to be detect attack according to signature-based detection we inject it by using DARPA dataset the result is shown in Figure 4.15 and 4.16, it show that show that the total number of alert that events created and is equal to 3502 alerts, and the category of these

alert classified into 9 type which shown in figure 4.16.

However Spade cannot tell you about is this packet is an attack or misconfigured, it just concerns about the normal and abnormal activity after adding and configure anomaly preprocessor for snort, we inject both systems and see the result for detection the total number of alerts have been increased to 5054 alerts as shown in figure 4.23, another comparison from previous time and the ICMP detection percentage is increased to 51% this explains that there is an abnormal activity on passed packets, the result is shown that the high performance for system that with anomaly preprocessor.

Although this result of detection is grows the drawback to anomaly detection is an alarm is generated any time traffic or activity deviates from the defined “normal” traffic patterns or activity. This means it’s up to the security administrator to discover why an alarm was generated. Anomaly-based systems have a higher rate of false positives because alarms are generated any time a deviation from normal occurs. Also As stated earlier, the signature based techniques are good but has the obvious short comings like failure to detect novel attacks, increasing signature database etc.

In summary, the results show that the approach followed in this thesis is quite effective in detecting network attack but it’s also need to human intervention to discover and make sure about why this alarm is generate and created also to distinguish if this alarm is false or not.

## **Chapter Five**

# **CONCLUSION**

## CHAPTER FIVE

### Conclusion & Recommendation

#### 5 Conclusion

IDS can be deployed to increase security and control within a corporate computing environment, almost IDS use one of these two method for detection which is misuse detection or Anomaly detection, for secure the system's and both of them have their own limitations, for misuse (Signature) Detection the system only detect the attack signature that store in database signature file and anomaly detection also may sometime send false alarm about attack, also This research shown how to implement and design IDS system for your security needing, it proposes that how to integrate both IDS method (Signature and Anomaly) in the Snort software which is based on signature detection for anomaly method we use SPADE Statistical Packet Anomaly Detection Engine preprocessor, after that testing the system using DARPA dataset, to validate this system we also inject the same data set to the signature-based IDS system, and the result is come out with high value of detection for system with anomaly preprocessor.

The research is come out with the following point it must be kept in mind which can be followed to get the highly secured infrastructure

- Intrusion Detection system is no a solution for all security concerns, IDS perform an excellent job in process of detect, monitor and report and in order to have good security policy IDS need to integrate with IPS.
- Successful intrusion detection requires that a well-defined policy must be followed to ensure that intrusions and vulnerabilities, virus

outbreaks, etc. are handled according to corporate security policy guidelines

- IDS require human intervention when the attack to detecting and report even is blocked, the IDS system requires administrator intervention to prevent the occurrence of the same attack in future.

- For Signature detection method when you deploy snort is important to make sure that you are using the rule that is relevant and up to date, otherwise, the system will be much less efficient.

- If work on the anomaly based detection this can generate a large number of false positive and false negative alarm, this may decrease the performance due to a large number of false positive, but the system work on misuse detection method it's impossible to detect unknown attack, to solve this problem a hybrid IDS are created and develop which uses both anomaly and misuse based ids to find out the unknown attacks and to raise the detection rate and lower false positive and false negative.

- The challenging that could be faced when deploying the IDS is that when you compared with the other IDSs system, Detection rate and false alarm rate are the best-suited evaluation metric for IDS system, the detection rate is the total intrusion injected in traffic and false alarm rate is like false positive alarm, So for Future work it's trying to test the system with real Network traffic to get better result on validating the system.

## REFERENCE

- [1] J. P. Anderson, "Computer Security Threat Monitoring and Surveillance", Technical Report April 1980, <http://csrc.nist.gov/publications/history/ande80.pdf>
- [2] Hybrid Ids System Using Snort Okasha Eldow\*, Punit Lalwani\*\*, m.b.Potdar\*\*\* Gtu Pg School, Gandhinagar, Bhaskaracharya Institute For Space Applications And Geo-Informatics Bisag, Gandhinagar, Gujarat, India, Volume 7 • Number 2 March 2016 - Sept 2016 Pp. 16-19
- [3] Dinakara K (06CS6026) "Anomaly-based Network Intrusion Detection System" Computer Science and Engineering, Department of Computer Science and Engineering Indian Institute of Technology Kharagpur -721302, India (May 2008)
- [4] Kaur T, Kaur S. Comparative analysis of anomaly based and signature-based intrusion detection systems using PHAD and Snort.
- [5] Kenkre, Poonam Sinai, Anusha Pai, and Louella Colaco. "Real-time intrusion detection and prevention system." Proceedings of the 3rd International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA) 2014. Springer, Cham, 2015.
- [6] Drum R. IDS and IPS placement for network protection. CISSP (March 26, 2006). 2006 Mar.
- [7] D. J. Brown, B. Suckow, and T. Wang, "A Survey of Intrusion Detection Systems," 2002
- [8] White paper, "Intrusion Detection: A Survey," ch.2, DAAD19-01, NSF, 2002



- [9] K. Scarfone, P. Mell, "Guide to Intrusion Detection and Prevention Systems (IDPS)," NIST Special Publication 800-94, Feb. 2007.
- [10] Kurundkar, G. D., N. A. Naik, and S. D. Khamitkar. "Network intrusion detection using Snort." *International Journal of Engineering Research and Applications* 2.2 (2012): 1288-1296.
- [11] Guimaraes, Mario, and Meg Murray. "Overview of intrusion detection and intrusion prevention." *Proceedings of the 5th annual conference on Information security curriculum development*. ACM, 2008.
- [12] Rehman, R. U. (2003). *Intrusion detection systems with Snort: advanced IDS techniques using Snort, Apache, MySQL, PHP, and ACID*. Prentice Hall Professional.
- [13] Roesch, M. (1999, November). *Snort: Lightweight intrusion detection for networks*. In *Lisa* (Vol. 99, No. 1, pp. 229-238).
- [14] Kumar, Vinod, and Om Prakash Sangwan. "Signature-based intrusion detection system using SNORT." *International Journal of Computer Applications & Information Technology* 1.3 (2012): 35-41.
- [15] Megha, "Gupta Hybrid Intrusion Detection System: Technology and Development" *International Journal of Computer Applications* (0975 – 8887) Volume 115 – No. 9, April (2015)
- [16] Garg, Mukta. "Intrusion Detection System in Campus Network: SNORT—the most powerful Open Source Network Security Tool." *volume 1, issue 4, October (2014)*.
- [17] Gupta, Abhishek, et al. "Network intrusion detection types and analysis of their tools." *International Journal of Engineering* 2.1

(2012): 63-69.

- [18] Rani, Suman, and Vikram Singh. "SNORT: an open source network security tool for intrusion detection in campus network environment." *International Journal of Computer Technology and Electronics Engineering* 2.1 (2012): 137-142.
- [19] T. H. Ptacek and T. N. Newsham, "Insertion, Evasion and Denial of Service: Eluding Network Intrusion Detection", *Secure Networks, Inc.*, Jan. 1998. <http://www.aciri.org/vern/Ptacek-NewshamEvasion-98.ps>.
- [20] Anomaly-based network intrusion detection: Techniques, systems and challenges by P. Garcíateodoro, J. Díaz-Verdejo, G. Maciá-Fernández & E. Vaázquez in *computers & security* 28 ( 2 0 0 9 ) Elsevier PP.18-28
- [22] Rehman, r.u., *intrusion detection systems with SNORT: advanced ids techniques using snort, Apache, MySQL, PHP, and acid*. 2003: prentice hall professional.
- [23] Sekar, R. et al. *A High-Performance Network Intrusion Detection System*. Proceedings of the 6th ACM conference on Computer and Communications Security. Singapore, November 1999. New York: ACM Press, 1999.
- [24] Barruffi, Rosy, Michela Milano, and Rebecca Montanari. "Planning for Security Management". *IEEE Intelligent Systems & Their Applications*. Los Alamitos, CA: IEEE Computer Society, Publications Office.
- [25] Shen, Y. Peggy et al. *Attack Tolerant Enhancement of Intrusion Detection Systems*. Proceedings of MILCOM 2000, 21st Century

Military Communications Conference. Los Angeles, CA, 22-25  
New York: Institute of Electrical and Electronics Engineers, 2001.

[26] <https://www.ll.mit.edu/r-d/datasets/1999-darpa-intrusion-detection-evaluation-data-set>

# Appendix A

## A1: Installing and Configuring the Necessary Prerequisites

In order for a BASE to function properly, install and configure a back-end database (MySQL) to store the Snort alerts. also install Apache and compiled Snort with MySQL support. PHP and couple of PHP add-ons are also needed. ADODB is an object-oriented PHP library used to interface to the database.

### Required Package

```
yum install -y gcc flex bison zlib * libxml2 libpcap * pcre * tcpdump git  
libtool curl man make daq
```

```
yum groupinstall - "Development Tools"
```

Prepare separate installation files

libdnet-1.12.tgz SV: Truong Ngoc Hien, Ho Duy Quang

libdnet-1.12-6.el6.x86\_64.rpm Class: 23CCAN04

libdnet-devel-1.12-6.el6.x86\_64.rpm

### MySQL

```
Shell > yum install mysql
```

```
Shell > yum install dlevel
```

```
Shell > yum install php mysql
```

(here will download and install MYSQL and a necessary packet that needed )

### Apache2

It is the most widely used in [web servers](#) software. Download and unpack Apache httpd server version 2.2.0 from the Apache httpd server website. To install apache2 follow the steps: must unpack it by using this command:

```
Shell > tar -zxvf httpd.tar.gz
```

```
Shell > cd httpd
```

```
Shell > ./configure
```

```
Shell > make
```

```
Shell > make install
```

## PHP

That is especially suited for b development and can be embedded into HTML.

```
Shell > tar -xvf php-4.4.2.tar
```

```
Shell > cd php-4.4.2.tar
```

Then configure PHP using the following commands:

```
Shell > ./configure --with-mysql --with-apxs2 =/usr/local/apache2/bin/apxs --with-gd --with-zlib
```

```
Shell > make
```

```
Shell > make install
```

After the installation edit the httpd.conf file (/usr/local/apache2/conf/httpd.conf) with our text editor. We add the following line to httpd.conf.

```
Include conf.d/*.conf
```

This allows us to create a specific configuration file for each module that we install, for instance php.conf. Now, we create a directory in our apache directory called conf.d.

```
mkdir /usr/local/apache2/conf.d
```

```
cd /usr/local/apache2/conf.d
```

We make a file called php.conf located at /usr/local/apache2/conf.d/php.conf with the following contents:

```
# PHP Configuration for Apache

# Load the apache module

LoadModule php4_module modules/libphp4.so

# Cause the PHP interpreter handle files with a .php extension.

<Files *.php>
SetOutputFilter PHP
SetInputFilter PHP
LimitRequestBody
9524288 </Files>

AddType application/x-httpd-php .php AddType application/
x-httpd-php-source .phps

# Add index.php to the list of files that will be served as directory

# Indexes.
```

We could have just inserted the above in the httpd.conf file, and omit the

conf.d step but this approach is a much cleaner way to do it

## Testing PHP and Apache

First we need to turn on and set to start the services will need. For that we do the following:

```
Shell > chkconfig mysql on
```

```
Shell > chkconfig httpd on
```

```
Shell > service httpd start
```

```
Shell >service mysqld start
```

## Snort

The following is the installation procedures on Centos Linux operating system:

```
shell> tar -xzvf snort-2.2.0 .tar.gz
```

```
shell> cd snort-2.2.0
```

```
shell> ./configure --with-mysql=/usr/local/mysql
```

#the above connect the snort with our database that will create it later

```
shell> make
```

```
shell> make install
```

## Install snort rule

```
shell> useradd snort
```

```
shell>groupadd -g snort snort
```

```
shell> mkdir /etc/snort
```

```
shell> mkdir /var/log/snort
```

```
shell> cd /etc/snort/rules
```

```
shell> cp * /etc/snort
```

```
shell> cd ../etc
```

```
shell> cp snort.conf /etc/snort
```

```
shell> cp *.config /etc/snort
```

## Snort configuration file .

Then we need to modify the snort.conf file which is located in /etc/snort. We need to make the following changes:

```
vi /etc/snort/snort.conf
```

```
ipvar HOME_NET nà> ipvar HOME_NET 192.168.x.x
```

```
ipvar EXTERNAL_NET any> ipvar EXTERNAL_NET! $  
HOME_NET
```

```
var SO_RULE_PATH ../so_rules> var SO_RULE_PATH / etc / snort /  
so_rules
```

```
var PREPROC_RULE_PATH ../preproc_rules> var  
PREPROC_RULE_PATH / etc/snort/preproc_rules
```

```
var WHITE_LIST_PATH ../rules> var WHITE_LIST_PATH / etc / snort  
/ rules
```

```
var BLACK_LIST_PATH ../rules> var BLACK_LIST_PATH / etc /  
snort / rules
```

Now we need to tell snort to log to MySQL. We go down to the output section and un-comment the following line. The password we create here is needed later on when we set up the Snort user in mysql.

**(output database: log, mysql, user=snort password=snort dbname=snort host=localhost)**

## Setting up the database in MySQL

Following are instructions for setting up the database in MySQL to be used by Snort. For the snort user, the password is what we put in the output section of the snort.conf in section 2.5.

```
mysql  
mysql> SET PASSWORD FOR root@localhost=PASSWORD('password');  
>Query OK, 0 rows affected (0.25 sec)  
mysql> create database snort;  
>Query OK, 1 row affected (0.01 sec)  
mysql> grant INSERT,SELECT on root.* to snort@localhost;  
>Query OK, 0 rows affected (0.02 sec)  
mysql> SET PASSWORD FOR snort@localhost=PASSWORD('password_from_snort.conf');  
  
>Query OK, 0 rows affected (0.25 sec)  
mysql> grant CREATE, INSERT, SELECT, DELETE, UPDATE on snort.* to snort@localhost;  
>Query OK, 0 rows affected (0.02 sec)  
mysql> grant CREATE, INSERT, SELECT, DELETE, UPDATE on snort.* to snort;  
>Query OK, 0 rows affected (0.02 sec)  
mysql> exit
```

Then we have to create tables for our database so we will execute this command :

```
shell> Mysql -u root -p < ~/snortinstall/snort-4.3.0/schemas/create_mysql  
snort
```

Enter password: the MySQL root password

Now we need to check and make sure that the Snort DB was created correctly.

As we see in the databases we have snort database that we have been creating it which include many 16 rows in the set . that means we have created snort database correctly.

```
mysql -p
>Enter password:
mysql> SHOW DATABASES;
(You should see the following)
+-----+
| Database
+-----+
| mysql
| Snort
| test
+-----+
3 rows in set (0.00 sec)
```

```
mysql> use snort
>Database changed
mysql> SHOW TABLES;
+-----+
| Tables_in_snort
+-----+
| data
| detail
| encoding
| event
| icmphdr
| iphdr
| opt
| reference
| reference_system
| schema
| sensor
| sig_class
| sig_reference
| signature

| tcphdr
| udphdr
+-----+
16 rows in set (0.00 sec)
exit;
```

## A2: Installing and Configuring BASE

### Downloading and Installing BASE

To install BASE, first, we go to our snort download directory. Then type “yum install php-gd”. This will install gd for proper graphing in the BASE. This will ask for the following, choose Y (YES)

Transaction Listing:

Install: php-gd.i386 0:4.3.10-3.2

Is this ok [y/N]: y

After we download it we will unpack it and make install and configure as we do last time.

### Configuring BASE



To configure BASE, we go to our download directory and do the following:

```
shell> cp base-1.3.8.tar.gz /var/www/html
shell> cd /var/www/html
shell> tar -xzf base-1.3.8.tar.gz
shell> cd base-1.3.8.tar.gz
shell> cp base_conf.php.dist base_conf.php
shell> cd /
shell> cp /var/www/html/base-1.2 /usr/local/apache2/htdocs/
```

Then we edit the “base\_conf.php” file in /usr/local/apache2/htdocs/ and insert the following perimeters:

```
$BASE_urlpath = "/base";
$DBlib_path = "/usr/local/apache2/htdocs/adodb";
$DBtype = "mysql";
$alert_dbname = "snort";
$alert_host = "localhost";
$alert_port = "";
$alert_user = "snort";
$alert_password = "password_from_snort_conf";
$archive_dbname = "snort";
$archive_host = "localhost";
$archive_port = "";
$archive_user = "snort";
$archive_password = " password_from_snort_conf ";
```

Now we should have a functional BASE ready to use. Open a web browser and if the browser is on the localhost, type <http://localhost/base-1.3.8> to begin using the GUI to view and manage alerts. The page we see is as below (Figure 3)

## Basic Analysis and Security Engine (BASE)

- Today's alerts:	unique	listing	Source IP	Destination IP
- Last 24 Hours alerts:	unique	listing	Source IP	Destination IP
- Last 72 Hours alerts:	unique	listing	Source IP	Destination IP
- Most recent 10 Alerts:	any protocol	TCP	UDP	ICMP
- Last Source Ports:	any protocol	TCP	UDP	
- Last Destination Ports:	any protocol	TCP	UDP	
- Most Frequent Source Ports:	any protocol	TCP	UDP	
- Most Frequent Destination Ports:	any protocol	TCP	UDP	
- Most frequent 10 Addresses:	Source	Destination		
- Most recent 10 Unique Alerts				
- Most frequent 5 Unique Alerts				

Queried on: Sat, August 05, 2016 11:31:05  
 Database: alert@basehost : Schema Version: 107  
 Time Utilized: no alerts detected

[Search](#)  
[Graph Alert Data](#)  
[Graph Alert Detection Time](#)

---

Sensors/Total: 0 / 0  
 Unique Alerts: 0  
 Categories: 0  
 Total Number of Alerts: 0

- Src IP adds: 0
- Dest. IP adds: 0
- Unique IP lists: 0
- Source Ports: 0
- TCP (0) UDP (0)
- Dest Ports: 0
- TCP (0) UDP (0)

**Traffic Profile by Protocol**

TCP (0%)

---

UDP (0%)

---

ICMP (0%)

---

Portscan Traffic (0%)

---

---

Alert Group Maintenance | Cache & Status | Administration

BASE 1.3.8 (jodie) by Kevin Johnson and the BASE Project Team  
 Built on ACID by Roman Danyliw

[Loaded in 1 seconds]

Figure: BASE Main Page

## Appendix B

Include the signature detection rules

```
# -----
```

```
# LOCAL RULES
```

```
# -----
```

```
# This file intentionally does not come with signatures. we Put our local signatures rules
```

```
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"DDOS icmp possible communication"; icmp_id:0; itype:0; content:"ABCDEFGHJKLMNPQRSTUVWXYZ"; reference:arachnids,425; reference:cve,2000-0138; classtype:attempted-dos; sid:222; rev:3;)
```

```
alert tcp $EXTERNAL_NET 53 -> $HOME_NET :1024 (msg:"MISC source port 53 to <1024"; flow:stateless; flags:S,12; reference:arachnids,07; classtype:bad-unknown; sid:504; rev:7;)
```

```
alert udp $EXTERNAL_NET any -> $HOME_NET any (msg:"MISC Large UDP Packet"; dsize:>400; reference:arachnids,247; classtype:bad-unknown; sid:521; rev:2;)
```

```
alert tcp $EXTERNAL_NET any <> $HOME_NET 0 (msg:"BAD-TRAFFIC tcp port 0 traffic"; flow:stateless; classtype:misc-activity; sid:524; rev:8;)
```

```
alert udp $EXTERNAL_NET any <> $HOME_NET 0 (msg:"BAD-TRAFFIC udp port 0 traffic"; reference:bugtraq,576; reference:cve,1999-0675; reference:nessus,10074; classtype:misc-activity; sid:525; rev:9;)
```

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"BAD-TRAFFIC data in TCP SYN packet"; flow:stateless; dsize:>6; flags:S,12; reference:url,www.cert.org/incident_notes/IN-99-07.html; classtype:misc-activity; sid:526; rev:11;)
```

```
alert tcp any any -> any any (msg: "BAD TRAFFIC SAME SRC/DST IP ADDRESS"; sameip ;reference:bugtraq,2666 ; reference: cve,1999-0016 ; classtype: bad-unknown ; sid: 527 ; rev : 8 ;)
```

```
alert ip any any <> 127.0.0.0/8 any (msg:"BAD-TRAFFIC loopback traffic"; reference:url,rr.sans.org/firewall/egress.php; classtype:bad-unknown; sid:528; rev:6;)
```

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 80 (flags: S; msg:"hping flood possible dos"; flow: stateless; threshold: type both, track by_src, count 70, seconds 10; sid:10001;rev:1;)
```

## Appendix C

This Appendix include snort configuration file

```
#####  
# This file contains a sample snort configuration.  
# You should take the following steps to create your own custom configuration:  
#  
# 1) Set the network variables.  
# 2) Configure the decoder  
# 3) Configure the base detection engine  
# 4) Configure dynamic loaded libraries  
# 5) Configure preprocessors  
# 6) Configure output plugins  
# 7) Customize your rule set  
#####  
# Step #1: Set the network variables. For more information, see README.variables  
#####  
# Setup the network addresses you are protecting  
var HOME_NET any  
# Set up the external network addresses. A good start may be "any"  
var EXTERNAL_NET any  
# List of DNS servers on your network  
var DNS_SERVERS $HOME_NET  
# List of SMTP servers on your network  
var SMTP_SERVERS $HOME_NET  
# List of web servers on your network  
var HTTP_SERVERS $HOME_NET  
# List of sql servers on your network  
var SQL_SERVERS $HOME_NET  
# List of telnet servers on your network  
var TELNET_SERVERS $HOME_NET  
# List of ports you run web servers on  
portvar HTTP_PORTS [80,2301,3128,7777,7779,8000,8008,8028,8080,8180,8888,9999]  
# List of ports you want to look for SHELLCODE on.
```

```

portvar SHELLCODE_PORTS !80
# List of ports you might see oracle attacks on
portvar ORACLE_PORTS 1521
# other variables, these should not be modified

var
[64.12.24.0/23,64.12.28.0/23,64.12.161.0/24,64.12.163.0/24,64.12.200.0/24,205.188.3.0/24,205.188.
5.0/24,205.188.7.0/24,205.188.9.0/24,205.188.153.0/24,205.188.179.0/24,205.188.248.0/24]
# Path to your rules files (this can be a relative path)
# Note for Windows users: You are advised to make this an absolute path,
# such as: c:\snort\rules
var RULE_PATH /etc/snort/rules
var SO_RULE_PATH /etc/snort/so_rules
var PREPROC_RULE_PATH /etc/snort/preproc_rules
#####
# Step #2: Configure the decoder. For more information, see README.decode
#####
# Stop generic decode events:
config disable_decode_alerts
# Stop Alerts on experimental TCP options
config disable_tcpopt_experimental_alerts
# Stop Alerts on obsolete TCP options
config disable_tcpopt_obsolete_alerts
# Stop Alerts on T/TCP alerts
config disable_tcpopt_tcp_alerts
# Stop Alerts on all other TCPOption type events:
config disable_tcpopt_alerts
# Stop Alerts on invalid ip options
config disable_ipopt_alerts
# Alert if value in length field (IP, TCP, UDP) is greater th elength of the packet
# config enable_decode_oversized_alerts
# Same as above, but drop packet if in Inline mode (requires enable_decode_oversized_alerts)
# config enable_decode_oversized_drops
# Configure IP / TCP checksum mode
config checksum_mode: all
# Configure maximum number of flowbit references. For more information, see README.flowbits
# config flowbits_size: 64

```

```

# Configure ports to ignore
# config ignore_ports: tcp 21 6667:6671 1356
# config ignore_ports: udp 1:17 53
#####

# Step #3: Configure the base detection engine. For more information, see README.decode
#####

# Configure PCRE match limitations
config pcre_match_limit: 1500
config pcre_match_limit_recursion: 1500

# Configure the detection engine See the Snort Manual, Configuring Snort - Includes - Config
config detection: search-method ac-bnfa max_queue_events 5

# Configure the event queue. For more information, see README.event_queue
config event_queue: max_queue 8 log 3 order_events content_length

# Configure Inline Resets. See README.INLINE
# config layer2resets: 00:06:76:DD:5F:E3

#####

# Step #4: Configure dynamic loaded libraries.
# For more information, see Snort Manual, Configuring Snort - Dynamic Modules
#####

# path to dynamic preprocessor libraries
dynamicpreprocessor directory /usr/local/lib/snort_dynamicpreprocessor/

# path to base preprocessor engine
dynamicengine /usr/local/lib/snort_dynamicengine/libsf_engine.so

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

# Step #5: Configure preprocessors
# For more information, see the Snort Manual, Configuring Snort - Preprocessors
#####

# Target-based IP defragmentation. For more information, see README.frag3
preprocessor frag3_global: max_fragments 65536
preprocessor frag3_engine: policy windows timeout 180

# Target-Based stateful inspection/stream reassembly. For more information, see README.stream5
preprocessor stream5_global: max_tcp 8192, track_tcp yes, track_udp no
preprocessor stream5_tcp: policy windows, use_static_footprint_sizes, ports client 21 22 23 25 42 53

```

```
79 80 109 110 111 113 119 135 136 137 139 143 110 111 161 445 513 514 691 1433 1521 2100 2301
3128 3306 6665 6666 6667 6668 6669 7000 8000 8080 8180 8888 32770 32771 32772 32773 32774
32775 32776 32777 32778 32779, ports both 443 465 563 636 989 992 993 994 995 7801 7702 7900
7901 7902 7903 7904 7905 7906 6907 7908 7909 7910 7911 7912 7913 7914 7915 7916 7917 7918
7919 7920
```

```
# preprocessor stream5_udp: ignore_any_rules
```

```
# performance statistics. For more information, see the Snort Manual, Configuring Snort -
Preprocessors - Performance Monitor
```

```
# preprocessor perfmonitor: time 300 file /var/snort/snort.stats pktcnt 10000
```

```
# HTTP normalization and anomaly detection. For more information, see README.http_inspect
```

```
preprocessor http_inspect: global iis_unicode_map unicode.map 1252
```

```
preprocessor http_inspect_server: server default \
```

```
  apache_whitespace no \
```

```
  ascii no \
```

```
    bare_byte no \
```

```
    chunk_length 500000 \
```

```
    flow_depth 1460 \
```

```
    directory no \
```

```
    double_decode no \
```

```
    iis_backslash no \
```

```
    iis_delimiter no \
```

```
    iis_unicode no \
```

```
    multi_slash no \
```

```
    non_strict \
```

```
    oversize_dir_length 500 \
```

```
    ports { 80 2301 3128 7777 7779 8000 8008 8028 8080 8180 8888 9999 } \
```

```
    u_encode yes \
```

```
    non_rfc_char { 0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 } \
```

```
    webroot no
```

```
# ONC-RPC normalization and anomaly detection. For more information, see the Snort Manual,
Configuring Snort - Preprocessors - RPC Decode
```

```
preprocessor rpc_decode: 111 32770 32771 32772 32773 32774 32775 32776 32777 32778 32779
no_alert_multiple_requests no_alert_large_fragments no_alert_incomplete
```

```
# Back Orifice detection.
```

```
preprocessor bo
```

```
# FTP / Telnet normalization and anomaly detection. For more information, see README.ftptelnet
```

```
preprocessor ftp_telnet: global encrypted_traffic yes check_encrypted inspection_type stateful
```

```
preprocessor ftp_telnet_protocol: telnet \
```

```

ayt_attack_thresh 20 \
normalize_ports { 23 } \
detect_anomalies

preprocessor ftp_telnet_protocol: ftp server default \
    def_max_param_len 100 \
    ports { 21 2100 } \
    ftp_cmds { USER PASS ACCT CWD SDUP SMNT QUIT REIN PORT PASV TYPE STRU
MODE } \
    ftp_cmds { RETR STOR STOU APPE ALLO REST RNFR RNT0 ABOR DELE RMD MKD
PWD } \
    ftp_cmds { LIST NLST SITE SYST STAT HELP NOOP } \
    ftp_cmds { AUTH ADAT PROT PBSZ CONF ENC } \
    ftp_cmds { FEAT OPTS CEL CMD MACB } \
    ftp_cmds { MDTM REST SIZE MLST MLSD } \
    ftp_cmds { XPWD XCWD XCUP XMKD XRMD TEST CLNT } \
    alt_max_param_len 0 { CDUP QUIT REIN PASV STOU ABOR PWD SYST NOOP } \
    alt_max_param_len 100 { MDTM CEL XCWD SITE USER PASS REST DELE RMD SYST
TEST STAT MACB EPSV CLNT LPRT } \
    alt_max_param_len 200 { XMKD NLST ALLO STOU APPE RETR STOR CMD RNFR HELP } \
    alt_max_param_len 256 { RNT0 CWD } \
    alt_max_param_len 400 { PORT } \
    alt_max_param_len 512 { SIZE } \
    chk_str_fmt { USER PASS ACCT CWD SDUP SMNT PORT TYPE STRU MODE } \
    chk_str_fmt { RETR STOR STOU APPE ALLO REST RNFR RNT0 DELE RMD MKD } \
    chk_str_fmt { LIST NLST SITE SYST STAT HELP } \
    chk_str_fmt { AUTH ADAT PROT PBSZ CONF ENC } \
    chk_str_fmt { FEAT OPTS CEL CMD } \
    chk_str_fmt { MDTM REST SIZE MLST MLSD } \
    chk_str_fmt { XPWD XCWD XCUP XMKD XRMD TEST CLNT } \
    cmd_validity MODE < char ASBCZ > \
    cmd_validity STRU < char FRP > \
    cmd_validity ALLO < int [ char R int ] > \
    cmd_validity TYPE < { char AE [ char NTC ] | char I | char L [ number ] } > \
    cmd_validity MDTM < [ date nnnnnnnnnnnnnn[.n[n[n]]] ] string > \
    cmd_validity PORT < host_port >

preprocessor ftp_telnet_protocol: ftp client default \
    max_resp_len 256 \

```



```

bounce yes \
telnet_cmds no
# SMTP normalization and anomaly detection. For more information, see README.SMTP
preprocessor smtp: ports { 25 587 691 } \
inspection_type stateful \
normalize_cmds \
normalize_cmds { EXPN VRFY RCPT } \
alt_max_command_line_len 260 { MAIL } \
alt_max_command_line_len 300 { RCPT } \
alt_max_command_line_len 500 { HELP HELO ETRN } \
alt_max_command_line_len 255 { EXPN VRFY }
# Portscan detection. For more information, see README.sfportscan
preprocessor sfportscan: proto { all } memcap { 10000000 } sense_level { low }
# ARP spoof detection. For more information, see the Snort Manual - Configuring Snort -
Preprocessors - ARP Spoof Preprocessor
# preprocessor arpspoof
# preprocessor arpspoof_detect_host: 192.168.40.1 f0:0f:00:f0:0f:00
# SSH anomaly detection. For more information, see README.ssh
preprocessor ssh: server_ports { 22 } \
    max_client_bytes 19600 \
    max_encrypted_packets 20 \
    enable_respoverflow enable_ssh1crc32 \
    enable_sroverflow enable_protomismatch
# SMB / DCE-RPC normalization and anomaly detection. For more information, see
README.dcerpc2
preprocessor dcerpc2: memcap 102400, events [co ]
preprocessor dcerpc2_server: default, policy WinXP, \
    detect [smb [139,445], tcp 135, udp 135, rpc-over-http-server 593], \
    autodetect [tcp 1025:, udp 1025:, rpc-over-http-server 1025:], \
    smb_max_chain 3
# DNS anomaly detection. For more information, see README.dns
preprocessor dns: ports { 53 } enable_rdata_overflow
# SSL anomaly detection and traffic bypass. For more information, see README.ssl
preprocessor ssl: ports { 443 465 563 636 989 992 993 994 995 7801 7702 7900 7901 7902 7903
7904 7905 7906 6907 7908 7909 7910 7911 7912 7913 7914 7915 7916 7917 7918 7919 7920 },
trustservers, noinspect_encrypted
#####

```

```

# Step #6: Configure output plugins
# For more information, see Snort Manual, Configuring Snort - Output Modules
#####
# syslog
# output alert_syslog: LOG_AUTH LOG_ALERT
# pcap
# output log_tcpdump: tcpdump.log
# database
# output database: alert, <db_type>, user=<username> password=<password> test dbname=<name>
host=<hostname>
# output database: log, <db_type>, user=<username> password=<password> test dbname=<name>
host=<hostname>
output database: log, Mysql, user=snort password=abc123 dbname=snort host=localhost
# unified2
output unified2: filename snort.log, limit 128
# prelude
# output alert_prelude
# metadata reference data. do not modify these lines
include classification.config
include reference.config
#####
# Step #7: Customize your rule set
# For more information, see Snort Manual, Writing Snort Rules
#####
# site specific rules
include $RULE_PATH/local.rules
include $RULE_PATH/exploit.rules
include $RULE_PATH/ftp.rules
include $RULE_PATH/telnet.rules
include $RULE_PATH/rpc.rules
include $RULE_PATH/rservices.rules
include $RULE_PATH/dos.rules
include $RULE_PATH/ddos.rules
include $RULE_PATH/dns.rules
include $RULE_PATH/web-cgi.rules
include $RULE_PATH/web-coldfusion.rules

```

```
include $RULE_PATH/web-iis.rules
include $RULE_PATH/web-frontpage.rules
include $RULE_PATH/web-misc.rules
include $RULE_PATH/web-client.rules
include $RULE_PATH/web-php.rules
include $RULE_PATH/sql.rules
include $RULE_PATH/x11.rules
include $RULE_PATH/netbios.rules
include $RULE_PATH/misc.rules
include $RULE_PATH/attack-responses.rules
include $RULE_PATH/oracle.rules
include $RULE_PATH/mysql.rules
include $RULE_PATH/smtp.rules
include $RULE_PATH/imap.rules
include $RULE_PATH/pop2.rules
include $RULE_PATH/pop3.rules
include $RULE_PATH/nntp.rules
include $RULE_PATH/backdoor.rules
include $RULE_PATH/snmp.rules
include $RULE_PATH/icmp.rules
include $RULE_PATH/tftp.rules
include $RULE_PATH/scan.rules
include $RULE_PATH/finger.rules
include $RULE_PATH/web-attacks.rules
include $RULE_PATH/shellcode.rules
include $RULE_PATH/policy.rules
include $RULE_PATH/info.rules
include $RULE_PATH/icmp-info.rules
include $RULE_PATH/virus.rules
include $RULE_PATH/chat.rules
include $RULE_PATH/multimedia.rules
include $RULE_PATH/p2p.rules
include $RULE_PATH/spyware-put.rules
include $RULE_PATH/specific-threats.rules
#include $RULE_PATH/voip.rules
include $RULE_PATH/other-ids.rules
```

```
include $RULE_PATH/bad-traffic.rules
# decoder and preprocessor event rules
# include $PREPROC_RULE_PATH/preprocessor.rules
# include $PREPROC_RULE_PATH/decoder.rules
# dynamic library rules
# include $SO_RULE_PATH/bad-traffic.rules
# include $SO_RULE_PATH/chat.rules
# include $SO_RULE_PATH/dos.rules
# include $SO_RULE_PATH/exploit.rules
# include $SO_RULE_PATH/imap.rules
# include $SO_RULE_PATH/misc.rules
# include $SO_RULE_PATH/multimedia.rules
# include $SO_RULE_PATH/netbios.rules
# include $SO_RULE_PATH/nntp.rules
# include $SO_RULE_PATH/p2p.rules
# include $SO_RULE_PATH/smtp.rules
# include $SO_RULE_PATH/sql.rules
# include $SO_RULE_PATH/web-activex.rules
# include $SO_RULE_PATH/web-client.rules
# include $SO_RULE_PATH/web-misc.rules
# Event thresholding or suppression commands. See threshold.conf
include threshold.conf
```

## Appendix D

Extract Spade compressed file and navigate inside the file and copy the content of the folder to the top of snort distribution directory.

Commands :

```
Shell> tar -zxvf spade
```

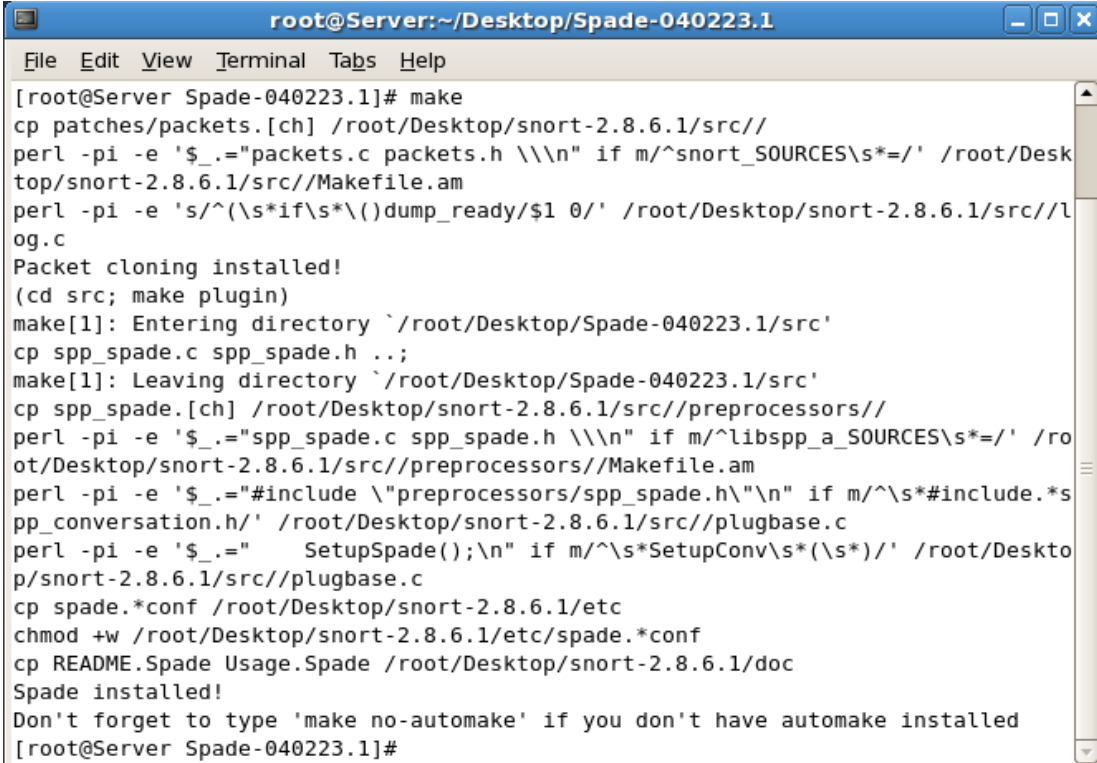
```
Shell> cd patches
```

```
Shell> gedit snort.conf.patch
```

#remove the snort.conf patch line to be adding latter

```
Shell> cd ..
```

```
Shell> make
```



```
root@Server:~/Desktop/Spade-040223.1
File Edit View Terminal Tabs Help
[root@Server Spade-040223.1]# make
cp patches/packets.[ch] /root/Desktop/snort-2.8.6.1/src//
perl -pi -e '$_="packets.c packets.h \\n" if m/^snort_SOURCES\s*=/ /root/Desktop/snort-2.8.6.1/src//Makefile.am
perl -pi -e 's/^(\\s*if\\s*\\()dump_ready/$1 0/' /root/Desktop/snort-2.8.6.1/src//log.c
Packet cloning installed!
(cd src; make plugin)
make[1]: Entering directory `/root/Desktop/Spade-040223.1/src'
cp spp_spade.c spp_spade.h ..;
make[1]: Leaving directory `/root/Desktop/Spade-040223.1/src'
cp spp_spade.[ch] /root/Desktop/snort-2.8.6.1/src//preprocessors//
perl -pi -e '$_="spp_spade.c spp_spade.h \\n" if m/^libspp_a_SOURCES\s*=/ /root/Desktop/snort-2.8.6.1/src//preprocessors//Makefile.am
perl -pi -e '$_="#include "preprocessors/spp_spade.h"\\n" if m/^\\s*#include.*spp_conversation.h/' /root/Desktop/snort-2.8.6.1/src//plugbase.c
perl -pi -e '$_=" SetupSpade();\\n" if m/^\\s*SetupConv\\s*(\\s*)/' /root/Desktop/snort-2.8.6.1/src//plugbase.c
cp spade.*conf /root/Desktop/snort-2.8.6.1/etc
chmod +w /root/Desktop/snort-2.8.6.1/etc/spade.*conf
cp README.Spade Usage.Spade /root/Desktop/snort-2.8.6.1/doc
Spade installed!
Don't forget to type 'make no-automake' if you don't have automake installed
[root@Server Spade-040223.1]#
```

After run make command if everything all-right the spade will be installed successfully.

After the we have to add to snort configuration file by adding spade log file path

```
preprocessor spade: logfile=/var/log/spade/spade.log
```

```
statefile=/var/log/spade/state.rcv cpfreq=25000 dest=alert adjdest=none
```

And then adding the home network for spade, here is set to any network class.

```
preprocessor spade-homenet: any
```

You now need to set up some detectors for SPADE to work with. Detectors are

set up with the following line, and the options detailed in the table below.

```
preprocessor spade-detect: { <optionname>=<value> }
```

The you have to include *spade.conf* file inside snort configuration file.

### **SPADE configuration file**

*# Example configuration file for Spade v021026.1 and later*

*# use this as your snort config file (-c option) to run Snort Spade-only*

*# include it in your snort config file or put lines of this form in it*

*# set this to a directory Spade can read and write to store its files*

*var SPADEDIR.*

*# see the Usage.Spade file for the full meaning of and all the options*

*# available for all these lines*

*# This is the main Spade configuration line; it must appear first.*

*# Here are some options for this line:*

*# + dest: the Snort facility that the Spade output should go to*

*# (alert, log, or both)*

*# + state file: where Spade's persistent data is stored*

*# + logfile: where Spade will store information about its run*

*# + Exports, Xdips, Xsips, Xsports: like below but with global application*

```
preprocessor spade: dest=alert logfile=$SPADEDIR/spade.log
statefile=$SPADEDIR/spade.rcv
```

*# This line sets up your Spade homenet. Set this to the network that is*

*# connecting to the larger network at the point Spade is running.*

*# It is important to configure this line.*

*# Your networks should be like [10.0.0.0/8,192.168.0.0/16] or space separated*

```
preprocessor spade-homenet: any
```

*# Turn on some detectors with "spade-detect" lines. Each of these enables*

*# a certain type of detector for a certain type of packet. If you start to*

*# feel overwhelmed, use Xdports, Xdips, Xsips, and/or Xsports on the lines*

*# below to suppress reports you don't care about, and/or disable some of*

*# your detectors these that you care least about.*

```

#   These detect packets going to seemingly closed dest ports
#       You can add thresh=N to override the default reporting threshold.
preprocessor spade-detect: type=closed-dport tcpflags=synonly wait=3
preprocessor spade-detect: type=closed-dport tcpflags=weird thresh=0.5
#preprocessor spade-detect: type=closed-dport tcpflags=synack
#preprocessor spade-detect: type=closed-dport tcpflags=established
#preprocessor spade-detect: type=closed-dport tcpflags=teardown
#preprocessor spade-detect: type=closed-dport proto=udp wait=5
#preprocessor spade-detect: type=closed-dport to=nothome tcpflags=synonly wait=5
#preprocessor spade-detect: type=closed-dport to=nothome tcpflags=weird
#preprocessor spade-detect: type=closed-dport to=nothome tcpflags=synack
#preprocessor spade-detect: type=closed-dport to=nothome tcpflags=established
#preprocessor spade-detect: type=closed-dport to=nothome tcpflags=teardown
#preprocessor spade-detect: type=closed-dport to=nothome proto=udp wait=7
#   These detect packets going to a seemingly non-live IP
#preprocessor spade-detect: type=dead-dest tcpflags=synonly wait=2
preprocessor spade-detect: type=dead-dest tcpflags=weird wait=2
preprocessor spade-detect: type=dead-dest tcpflags=synack wait=2
#preprocessor spade-detect: type=dead-dest tcpflags=setup wait=2
preprocessor spade-detect: type=dead-dest tcpflags=established wait=5
preprocessor spade-detect: type=dead-dest tcpflags=teardown wait=2
preprocessor spade-detect: type=dead-dest proto=udp wait=2
preprocessor spade-detect: type=dead-dest proto=icmp icmptype=noterr wait=2
#preprocessor spade-detect: type=dead-dest proto=icmp icmptype=err wait=2
#   These detect unusual use of a dest port by a source IP
#       You can add thresh=N to override the default reporting threshold.
#preprocessor spade-detect: type=odd-dport proto=tcp wait=2
#preprocessor spade-detect: type=odd-dport proto=udp wait=5
#preprocessor spade-detect: type=odd-dport from=nothome proto=tcp
#preprocessor spade-detect: type=odd-dport from=nothome proto=udp
#   These detect ICMP packets with an unusual type and code
#       You can add thresh=N to override the default reporting threshold.

```

```

preprocessor spade-detect: type=odd-typecode
preprocessor spade-detect: type=odd-typecode to=nothome
#   These detect unusual connections to a dest IP by a source IP when the
#   dest port has predictable dest IPs
#   You can add thresh=N to override the default reporting threshold.
#preprocessor spade-detect: type=odd-port-dest proto=tcp Xdports=80
#preprocessor spade-detect: type=odd-port-dest proto=udp Xdports=80
#preprocessor spade-detect: type=odd-port-dest from=nothome proto=tcp
Xdports=80
#preprocessor spade-detect: type=odd-port-dest from=nothome proto=udp
Xdports=80
# This line causes Spade to adjust the reporting threshold for a given
# detector automatically; repeat it for each detector that you want to apply
# it to
# Target is the target rate of alerts for normal circumstances
# (0.01= 1% or you can give it an hourly rate)
# After the first hour (or however long the period is set to with "obsper"),
# the initially configured reporting threshold is ignored
# To use this, you will need to an option of the form id=<label> to the
# spade-detect line of the detector you want to be adapted and set
# id=<label> below to match
# This mode is recommended for users getting started that are using absolute
# anomaly scores; relative score users might want it as well.
#preprocessor spade-adapt3: id=<label> target=0.01 obsper=60
# some other possible Spade config lines:
# offline threshold advising for a detector
#preprocessor spade-threshadvise: id=<label> target=200 obsper=24
# periodically report on the anom scores and count of packets seen by a detector
#preprocessor spade-survey: id=<label> surveyfile=$$SPADEDIR/survey.txt
interval=60
# print out certain all known stats about packet features
#preprocessor spade-stats: entropy uncondprob condprob

```