Design and Development of

Graphical User Interface for building Snort Rules

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Technology

in

Computer Science and Applications

Submitted By

Isha Singla

(601003009)

Under the supervision of:

Ms. Sanmeet Kaur

Assistant Professor

SMCA

School of Mathematics and Computer Applications

THAPAR UNIVERSITY

PATIALA – 147004

June 2012

Certificate

I hereby certify that the work which is being presented in the thesis entitled, *"Design and Development of Graphical User Interface for building Snort Rules"*, in partial fulfillment of the requirements for the award of degree of Master of Technology in *Computer Science and Application* submitted in School of Mathematics & Computer Applications Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of *Mrs. Sanmeet Kaur* and refers other researcher's work which are duly listed in the reference section.

The matter presented in the thesis has not been submitted for award of any other degree of this or any other University.

Isha Singla.
(Isha Singla)
601003009

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

Sanmeet Kaur
(Mrs. Sanmeet Kaur)
Assistant Professor
SMCA

Countersigned by

(Dr. S.S. Bhatia)
Head,
SMCA Department,
Thapar University,
Patiala.

(Dr. S. K. Mohapatra)
Dean (Academic Affairs),
Thapar University,
Patiala.

i

Acknowledgement

I would like to express my deepest appreciation to Mrs. Sanmeet Kaur, my mentor and thesis supervisor for her constant support and motivation. She had been instrumental in guiding me throughout the thesis with their valuable insights, constructive criticisms and interminable encouragement.

I am also thankful to Dr. S.S. Bhatia, Head, School of Mathematics and Computer Applications Department and Mr. Singara Singh, P.G. Coordinator for their constant support and encouragement.

I would like to thank all the faculty members and staff of the department who were always there at the need of the hour and provided with all the help and facilities, which I required, for the completion of this work.

I express my thanks to my family for their support and affection and for believing in me always. I also want to thank my colleagues, who have given me moral support and their relentless advice throughout the completion of this work.

Isha Singla

(601003009)

Abstract

Network Security is becoming an important issue for all the organizations, and with the increase in knowledge of hackers and intruders they have made many successful attempts

to bring down high-profile company networks and web services. With the recent advances in the field of network security a technique called Intrusion Detection System are develop to further enhance and make your network secure. It is a way by which we can protect our internal network from outside attack, and can take appropriate action if needed. Using intrusion detection methods, information can be collected from known types of attack and can be used to detect if someone is trying to attack the network.

Both open source and commercial tools are available for detecting intrusion in a network, many vulnerability assessment tools are also available in the market. Many techniques are there to detect intrusion in a network like signature matching, anomaly based and others.

The thesis starts with the introductive study of various kinds of attacks in the network and then different tools to protect network from various malicious activities are studied. Then the detailed analysis of IDS is carried out. IDS components, types were studied based upon different factors. After that Snort network based intrusion detection is studied and analyzed. Its rule making process is discussed. In the end live traffic is captured using Snort itself and using wireshark also and the traffic is analyzed with the help of Snort. Rules are created in Snort. Finally a front end is created to implement Snort in a GUI based environment. The front end is used in the live environment.

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1 Introduction to network security

With the introduction of the computer, the need for protecting files and other information stored on the computer became evident. This is especially the case of shared system, such as time sharing system, and the need is even more acute for systems that can be accessed over a public telephone network, data network, or the Internet.

Nowadays, security is considered as one of the most critical parameter for the acceptance of any networking technology. Information in transit must be protected from unauthorized release and modification, and the connection itself must be established and maintained securely. Detection of illegitimate traffic is one of the goals of communication security and seeks to prevent an eavesdropper from gaining any meaningful information about network users' behavior or objectives by observing the legitimate traffic on the network.

To protect the enterprise, security managers have deployed a variety of technologies. While these technologies are useful for defending corporate assets, they have limitations. For example, firewalls may be configured to block certain types of traffic, but attackers still find ways to exploit legitimate traffic types to mount their attacks. To ensure the security of network, the following major security objectives of any application have paramount importance [1].

- **Confidentiality:** It means that certain information is only accessible to those who have been authorized to access it.
- **Integrity:** Integrity guarantees that a message being transferred is never corrupted.
- **Availability**: Availability ensures the survivability of network services despite denial of service (DoS) attacks.
- **Authenticity:** Authenticity is essentially assurance that participants in communication are genuine and not impersonators.

- **Non-repudiation:** Non-repudiation ensures that the sender and the receiver of a message cannot deny that they have ever sent or received such a message.
- **Authorization**: Authorization is a process in which an entity is issued a credential by the trusted certificate authority.
- **Anonymity**: Anonymity means that all the information that can be used to identify the owner or the current user entity should be kept private and not distributed to other communicating parties.

1.2 Attacks in network

A useful means of classifying security attacks, used both in X800 and RFC2828, is in terms of passive attacks [2] and active attacks [2].

1.2.1 Passive attacks

Passive attacks [2] are in the nature of eavesdropping on, or monitoring of, transmissions. The goal of the opponent is to obtain information that is being transmitted. Two types of passive attacks are release of message contents and traffic analysis.

The release of message contents is easily understood. A telephone conversation, an electronic email message and a transferred file may contain sensitive or confidential information. One would like to prevent an opponent from learning the contents of these transmissions.

A second type of passive attack, traffic analysis is subtler. Suppose that there is a way of masking the contents of messages or other information traffic, so that opponents, even if they captured the message, could not extract the information from the message. The common technique for masking contents is encryption. If one had encryption protection in place, an opponent might still be able to observe the pattern of this message. The opponent could determine the location and identity of communicating hosts and could observe the frequency and length of messages being exchanged. The information might be useful in guessing the nature of the communication that was taking place.

Passive attacks are very difficult to detect because they do not involve any alteration of the data. Typically, the message traffic is not sent and received in an apparently normal fashion and the sender nor receiver is aware that a third party has read the message or observed the traffic pattern. However, is feasible to prevent the success of the attacks,

usually by means of encryption. Thus, the emphasis in dealing with passive attacks is on prevention rather than detection.

1.2.2 Active attacks

Active attacks [2] involve some modification of data stream or the creation of a false stream and can be subdivided into four categories:

- **Masquerade:** A masquerade takes place when one entity pretends to be a different entity. A masquerade attacks usually includes one of the other forms of active attacks. For example, authentication sequences can be captured and replayed after a valid authentication sequence has taken place, thus enabling an authorized entity with few privileges to obtain extra privileges by impersonating an entity that has those privileges.
- **Replay:** Replay involves the passive capture of a data unit and its subsequent retransmission to produce an unauthorized effect.
- **Modification of messages:** Modification of messages simply means that some portion of a legitimate message is altered or that messages are delayed or reordered, to produce an unauthorized effect. For example, a message meaning "Allow John Smith to read confidential files account" is modified to mean "Allow Fred Brown to read confidential file accounts".
- **Denial of service:** The denial of service prevents or inhibits the normal use of management of communications facilities. This attack may have a specific target, for example, an entity may suppress all messages directed to a particular destination (e.g. the security audit service). Another form of service denial disruption of an entire network, either by disabling the network or by overloading it with messages so as to degrade the performance.

1.3 Types of attacks

In terms of the relation intruder-victim, attacks are, internal, coming from own enterprise's employees or their business partners or customers and External, coming from outside, frequently via the internet**.**

- **SYN flooding:** "SYN flood" attack is a denial-of-service attack by sending a large amount of SYN packets to a network or a server [3]. The attack packets

usually have spoofed source addresses to hide the real attacking sources and also make defense much harder. For simplicity, from now on we call the victim of a SYN flood attack as a server. It is relatively easy to detect whether or not a server is under SYN flood DDoS attack by observing the existence of excessive amount of 1/2 open transmission control protocol (TCP) connections. However, filtering SYN attack packets is very hard: a filtering-based defense system must be able to detect individual incoming SYN attack packet based only on its packet header, even though all fields in the packet header can be arbitrarily modified by attackers.

- **Packet sniffers: P**acket sniffing [4] is a method of tapping each packet as it flows across the network; i.e., it is a technique in which a user sniffs data belonging to other users of the network. Packet sniffers can operate as an administrative tool or for malicious purposes. It depends on the user's intent. Network administrators use them for monitoring and validating network traffic.
- **Viruses:** A small piece of software that replicates itself on real programs and runs every time a program runs .Most can reproduce and attack other programs. Trojan horses: A computer program that contains hidden functions that can exploit the privileges of the user running the program . Can erase the disk, send your credit card numbers, and password to the hackers.
- **Spyware**: Spyware is a new type of potentially unwanted programs the goal of which is to monitor users' online behaviors without user consent [5]. Users infected with spyware commonly experience severely degraded reliability and performance such as increased boot time, sluggish feel, and frequent application crashes.

1.4 Phases of attacks

A professional attacker will thoroughly investigate the target systems, and ensure that everything is safe for the attack without being detected. Then they attack in a very structured manner while consistently monitoring the effect.

All successful intrusions share the following characteristic phases [6]:

- Reconnaissance, Assessment and Strategy

- Exploitation and Invasion
- Maintaining Access
- Operations

- **Reconnaissance, Assessment and Strategy:** Reconnaissance, or Recon, is the act of scoping out a target. This information gathering stage is the most important step an attacker takes, and all key information is considered. The Assessment and Strategy stage is the sorting of the gathered data to piece together an idea of what the hacker is attacking. Recon can go undetected for considerable lengths of time and the Assessment and Strategy stage is often completely undetectable, as it is usually done without contact with the target. These two stages are assessed together because Recon is the part of the act that involves interaction of some sort with the target, and the Assessment and Strategy stage is usually done remotely by reviewing the gathered data. To launch successful attacks, attackers need information about the topology of the network, about accessible network services, about software versions, about valid user/password credentials, and about anything else that will help them succeed in their effort.
- **Exploitation and Invasion**: Once an attacker has gathered enough information and has pieced together a reasonable amount of information about the network or system they are attacking, and have devised an initial plan of attack, it is then possible to begin the Exploitation and Invasion stage. At this point, the attacker uses the gathered knowledge and attempts to access the server through the channels that were found open. In this phase, the intruder has gained access to desired system facilities. Penetration and exploitation create a spiral of increasing intruder authority and a widening circle of compromise. For example, penetration at the user level is typically a means to find root-level vulnerabilities. User-level authorization is then employed to exploit those vulnerabilities to achieve root-level privileges. Finally, compromise of the weakest host in a networked system allows that host to be used as a stepping-stone to compromise other more protected hosts. The success of this stage is mainly depends on the time spend on the recon stage and the experience of the attacker in well assigning the target host.

- **Maintaining Access:** Once an attacker has penetrated the network (or if the attack is an inside job) steps are usually taken to make future accesses easier to conduct. This often includes installing a back-door program, but sometimes may be something as simple as setting up a home base under a seldom-used account name or identifying a mis-configured user account with suitable permissions to use to regain entry. Attackers install tools and manipulate existing software on a system to maintain access to the machine on their own terms. They install backdoors, apply "rootkits" (the process of substituting binary executables with nasty variations), and sometimes even manipulate the underlying kernel itself to hide their evil deeds. Attackers also cover their tracks by hiding files, sniffers, network usage, and running processes. Finally, attackers often alter system logs, all in an attempt to make the compromised system appear normal.
- **Operations:** This is the most dangerous part of a penetration; the attacker has all the access required to carry out his/her plan. If it is a spy operation, data could be sent to a remote collection repository. While this process does not give a concrete damage, it can damage the trust on the organization, and this will be reflected to the finance of that organization. Sometimes, this financial effect could be so harmful that it could lead to the bankruptcy of that organization. If the attack is a system-mapping reconnaissance mission, existing levels of access may be used to compromise more systems on the network. This mapping process can be used to find out the weaknesses of the systems to gain access to them. Then, those compromised systems, then, can be used to create a distributed denial of service (DDoS) attack. Or, that attack may be the first step for a spy operation and the compromised system could be used to gain access to more important systems on the network that could not be directly accessed. Understanding the stages of attack process is central to effective defense. In fact, security administrators can take advantage of inherent flaws in the attack process to actually prevent attacks before they reach to its target. Just as attackers exploit vulnerabilities in the network to mount attacks, security administrators can exploit vulnerabilities in the attack process to protect themselves.

1.5 Network security tools

Similar to physical security, information security managers have utilized multiple technologies to keep their network safe. However, as an effect of the improvements in technology, networks are now connected to one or more outside networks – including, of course, the Internet. Hence, the corporations face with a wide range of threats. So, security managers are under a lot of pressure to prevent any penetration to the network perimeter. Similar to the physical security, there are numerous security tools to help security managers in setting up complex protection strategy plans for their computer systems. Mostly common ones are commented subsequently.

1.5.1 Firewalls

Firewall [7] is a hardware or software solution implemented within the network to enforce security policies by controlling network access. The traditional function of firewalls has evolved from the original function of protecting a network from unauthorized external access. Today's firewalls inspect and filter traffic arriving or departing a network by comparing packets to a set of rules and performing the matching rule action, which is accept or deny. In general, firewalls can offer data privacy (confidentiality), integrity, and availability.

A firewall is often seen as the first step toward a network security solution. Firewalls must be installed at the choke points to control network traffic and implement network security policy of the organization for its external network connections, especially for the Internet. Because many Internet-based services are inherently insecure, a firewall must help an organization to disable some services and restrict others according to the organizational security policy.

As it can be seen from Figure 1.1, firewall can act as a wall between the two networks. The person want access to the either network has to pass this wall before entering. A firewall is a system that is set up to control traffic flow between two networks. Firewall prevents what IDS detect

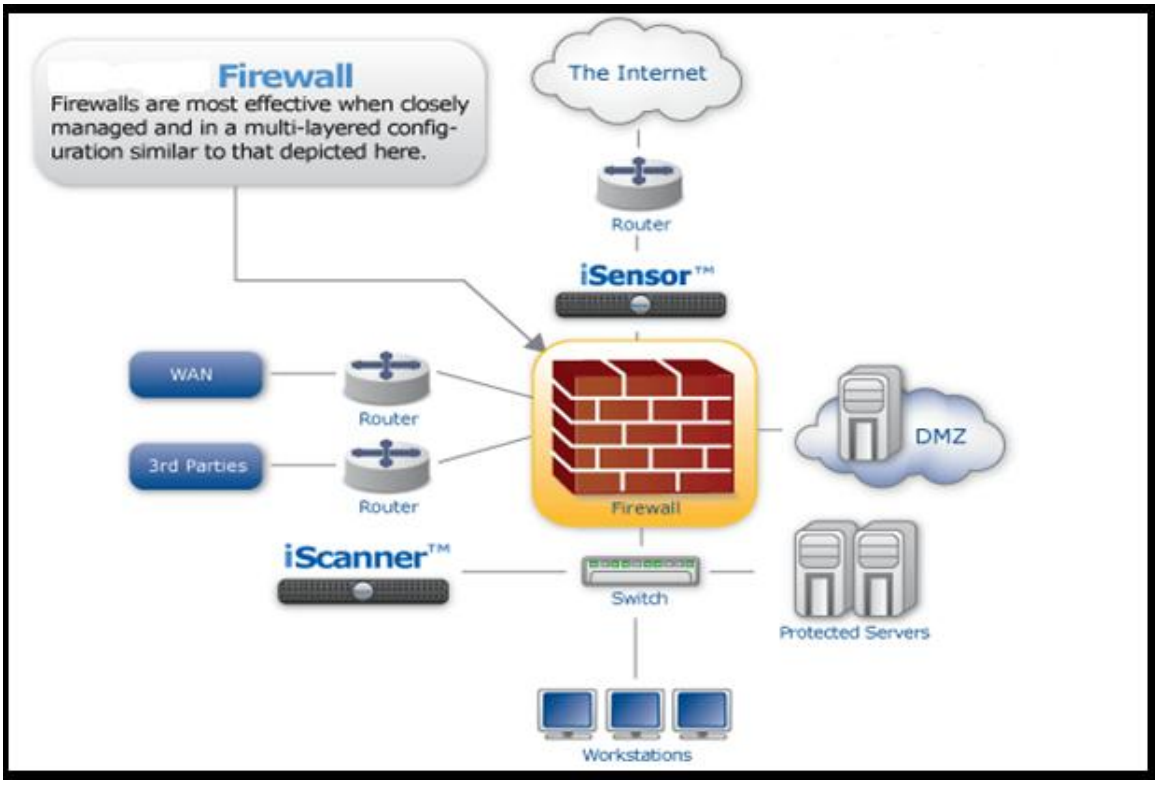


Fig 1.1: Firewall [7]

There are various firewall products but they are grouped into three major types based on their mechanisms: packet filtering, stateful inspection, and proxying.

- **Packet filtering**: Packet filtering is a mechanism that control which packets can go to and come from a network by examining their headers. There are no content-based decisions. The decision is solely based on the packet headers which include source address, destination address, type of traffic (such as TCP, UDP, ICMP), and characteristics of the transport layer communications sessions (such as source and destination ports). Packet filters are associated with interfaces therefore, the interface that the packet comes from or will go through can be restricted. Packet filter firewall can have rules such as; letting some hosts send email via SMTP (Simple Mail Transfer Protocol) or not letting any outside host connect to an internal host using Telnet.
- **Stateful inspection:** Another technology, stateful inspection, evolved from the need to accommodate certain features of the TCP/IP protocol suite. In essence, stateful inspection firewalls are packet filter firewalls with the connection status

awareness capability. This awareness is done by making a dynamic list of active connections between hosts, called state table. A packet that does not belong to an active connection and is not a connection request is refused by the firewall. A packet that belongs to an active connection is allowed through, bypasses the firewall rules, therefore optimizing the inspection process.

- **Proxying:** Proxying is a mechanism that provides all internal hosts the external (untrusted) network access while appearing that a single host accessing the outside. Since all connection to the external network be done by a single host, deep packet inspection is possible before passing packets to internal hosts. Proxying examines source address, destination address, protocol used, source port, destination port and also payload (content) of packets.

The advantages and disadvantages of firewall are given below:

- **Advantages of Firewalls:** Following are the advantages of Firewalls:
  i. Firewalls can stop non-legitimate traffic at first point.
  ii. Firewalls can filter protocols and services that are either not necessary or that cannot be adequately secured from exploitation.
  iii. A firewall can "hide" names of internal systems and internal network schema, thereby revealing less information to outside hosts.
  iv. Firewalls can concentrate extended logging of network traffic on one system.
- **Disadvantages of Firewalls:** Following are the disadvantages of Firewalls:
  i. Firewalls utilize manually configured set of rules to differentiate legitimate traffic from non-legitimate traffic,
  ii. Once a static policy is defined, the firewall can't react to a network attack – nor can it initiate effective counter-measures [8].
  iii. Most firewalls do not analyze the contents of the data packets that make up network traffic.
  iv. Firewalls cannot prevent attacks coming from Intranet [8].
  v. Filtering rules of the firewall are usually very simple, so firewall cannot prevent attack coming from application layer, and cannot prevent virus also [8].

1.5.2 Intrusion Detection System

A second layer in the perimeter defense is intrusion detection systems [9]. The purpose of an Intrusion Detection System is to detect both external attacks and internal misuse of computer and network resources or information residing in these resources. As it can be seen from Figure 1.2 below IDS protects the internal as well as external network from outside attack .In the physical analogy, an Intrusion detection system is equivalent to a video camera and motion sensor detecting unauthorized or suspicious activity and working with automated response. These devices, similar to firewalls, inspect incoming and outgoing network traffic. Unlike firewalls, however, they do not alter the traffic flow by dropping or passing certain packets. Rather, they look for malicious traffic that may be indicative of an attack or other misuse and log an alarm with specific data for administrative review. IDS can be divided into two general types known as anomaly detection and misuse detection [10]. The main challenge in intrusion detection is that of separating anomalous events from normal events.

An ideal Intrusion Detection System needs to satisfy two criteria:

- It must be able to correctly identify intrusions
- It must not identify legitimate action in a system environment as an intrusion.

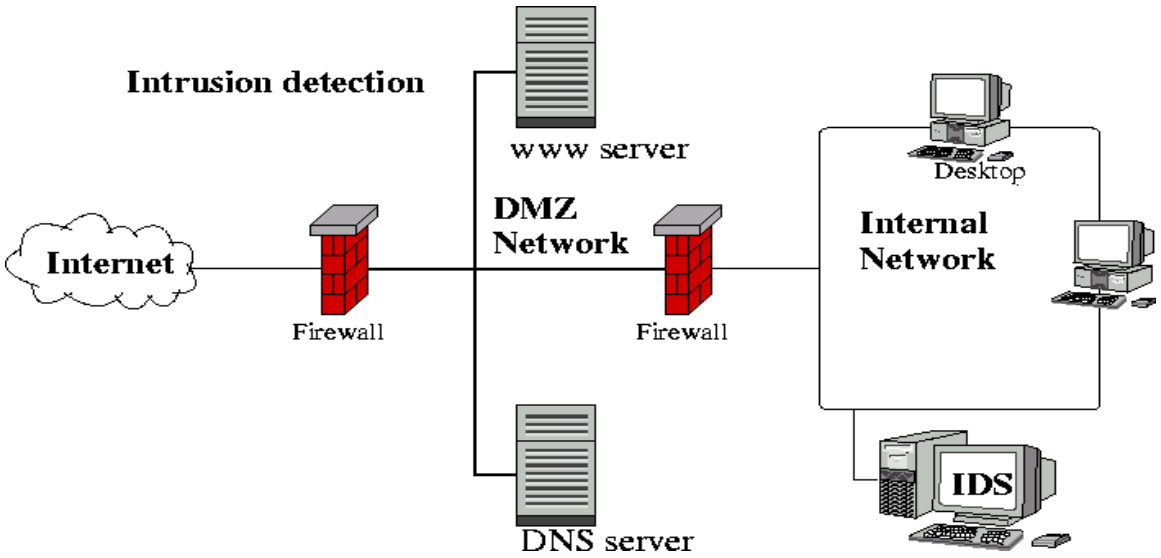


Fig 1.2: Intrusion Detection System [11]

The advantages and disadvantages of IDS are as follows:

- **Advantages of IDS :**Intrusion detection systems are advantages because of the following:
    i. The deployment of network-based IDSs has little performance impact upon an existing network,
    ii. Host-based IDSs are able to monitor events local to a host.
    iii. IDSs can allow security managers, regardless of their level of security expertise, to track security problems on their systems, initiating incident handling procedures.
- **Disadvantages of IDS:** Intrusion detection systems are disadvantageous because of the following reasons.
    i. IDSs have a tendency to generate "false positives". That is, they frequently generate alerts about an attack when none is taking place,
    ii. The only way to eliminate false positives would be to tune the system down to the point where it would also ignore real attacks – yielding "false negatives" – an obviously unacceptable approach.
    iii. IDS must be closely monitored and continually fine-tuned to the usage patterns and vulnerabilities discovered in its deployed environment. Such maintenance typically consumes a fair amount of administrative resources and effort,
    iv. A substantial amount of time may pass between the attack and the remediation, allowing the attacker to do irreversible damage in the meantime,
    v. Any IDS system that relies exclusively on documented attack profiles will be vulnerable to new, as-yet-undocumented attacks.
    vi. IDS is passive and reactive, its main goal is to detect attack actions, not to prevent them [8].
    vii. There are higher misinformation rate, it is difficult to make the detector and access control policy more exact and effective [8].

1.5.3 Honeypots

Honeypots [12] are highly flexible security tools with different applications for security. Unlike Firewalls or IDS, Honeypots do not solve a specific problem. Instead, they have multiple uses, such as prevention, detection, or information gathering. Honeypots only capture bad activity; any interaction with a honeypot is most likely unauthorized or malicious activity. Thus, honeypots collect small data sets having high value, as the data set contains only the attacks. This means it's much easier to analyze the data a honeypots collects and derives value from it. Honeypots are designed to:

- Divert an attacker from accessing critical systems.
- Collect information about the attacker's activity.
- Encourage the attacker to stay on the system long enough for security managers to respond.

Honeypots are highly flexible security tools with different applications for security. Unlike firewalls or IDSs, honeypots do not solve a specific problem. Instead, they have multiple uses, such as prevention, detection, or information gathering. There are various implementations but they all share the same concept: a security resource that should not have any production or authorized activity. Theoretically, a honeypot should see no traffic because it has no legitimate activity. This means any interaction with a honeypot is most likely unauthorized or malicious activity. Any connection attempts to a honeypot are most likely a probe, attack, or compromise. This is what a honeypot is, it is a security resource whose value lies in being probed, attacked, or compromised. Honeypots in a network should not affect critical network services and applications. Those series of characteristics distinguish honeypots clearly of other solutions of security.

Honeypots only capture bad activity; any interaction with a honeypot is most likely unauthorized or malicious activity. Thus, honeypots collect small data sets having high value, as the data set contains only the attacks. This means it's much easier (and cheaper) to analyze the data a honeypot collects and derives value from it

Honeypots can be divided into two types:

- **Production honeypots:** The purpose of production honeypots is to protect the organization; they directly increase resource security. Research honeypots, on the

other hand, have a primary purpose of gathering information on attackers. This information indirectly helps secure a network. In general, deploying production honeypots is easier and entails less risk, but such honeypots also capture less information about their attackers. Production honeypots help secure organizations in one of three ways—preventing, detecting, or helping them respond to attacks.

- **Research honeypots**: Research Honeypot is run by a volunteer who gathers information about the motives and tactics of intruders and attackers. It is used to research the threats organizations face, and to learn how to better protect against those threats. Research Honeypot is more complex to deploy and maintain, capture extensive information, and is used primarily by research, military, or government organizations. That information can be used in early warning and prediction, law enforcement or understanding trends in attacker activity. [12]

Some of the advantages and disadvantages associates to them are described as:

- **Advantages of Honeypots:** Various advantages of Honeypots are as the following:
  i. Honeypots lure attackers by presenting a more visible and apparently vulnerable resource than the enterprise network itself. Thus, they distract attackers from more valuable hosts on the network.
  ii. Honeypots are useful for detecting attacks, since they provide a single point for security managers to monitor for evidence of anomalous activity.
  iii. Honeypots are useful for forensics, since they can be specifically designed to retain data pertaining to an attack.
  iv. Honeypots allow in-depth examination of attacks during and after exploitation of the targets.
  v. Unlike most security technologies (such as IDS systems) honeypots work fine in encrypted or IPv6 environments.
  vi. Honeypots can provide early warning about new attack and exploitation trends.
- **Disadvantages of Honeypots:** Various disadvantages of Honeypots are as the following:
  i. Honeypots can only track and capture activity that directly interacts with them.

ii. They have limited view field, they capture activity which is directed towards them [13].
iii. Honeypots are not especially effective at attack prevention.
iv. Honeypots have the risk of being taken over by the attackers and being used to harm other systems.
v. A honeypot may offer as its highest value nothing more than a learning experience (that is, you may not catch any attackers).
vi. Honeypots are not able to prevent new attacks [13].

1.5.4 Network-based Antivirus systems

Fourth security technology being used by many organizations is network-based antivirus systems. Network-based antivirus [14] systems are solutions that are installed on a gateway between two networks to prevent the spreading of viruses across the network. The limitations of host-based antivirus software demonstrate the need for properly implemented network-based antivirus systems that allow security managers to deploy comprehensive antivirus protection faster, and guard against the rise of threats that endanger networks as they spread by exploiting known vulnerabilities.

Network-based antivirus systems are installed in the DMZ (Demilitarized Zone) in order to capture and compare incoming and outgoing packet contents to a database of known virus signatures. Unfortunately, network-based antivirus systems have to operate under much more difficult constraints than host-based antivirus systems have to. Files are transported over networks in the payload portions of packets, each containing a small chunk of the file. A typical packet payload on the Internet is approximately 1,500 bytes in length. However, many viruses are substantially longer than 1,500 bytes, and can exceed 100K bytes in length. As a result, it is not sufficient for network-based antivirus systems to simply scan each packet individually. If a virus is longer than 1,500 bytes, and the signature for the virus relies on patterns that occur in portions of the packet that are separated by more than 1,500 bytes, then a packet-by-packet scan will never detect it.

Therefore, several methods are used to detect viruses:

- **Signature-based:** Signatures are patterns of bytes that are unique to a particular virus. Signature-based antivirus products are composed of two key elements: a database that contains the signatures for known viruses, and a scanning engine

that compares files under investigation with the signatures in the database to detect a match indicating the presence of a virus. However, virus codes can be encrypted, which randomizes the code and makes it much harder to develop a signature. Moreover, there are polymorphic viruses, which actually modify themselves slightly at each replication, further complicating, and in some cases defeating the ability of antivirus vendors to develop signatures.

- **Heuristic scanning:** Heuristic scanning looks for patterns of known bad behavior, rather than looking for a specific virus signature. For example, some viruses read and write certain files or execute certain operations in a way that would never be found in legitimate programs. The sequences of operations that constitute these behaviors can also be used to develop so-called heuristic signatures, which enable antivirus engines to detect some viruses without an explicit signature.

  The advantages and disadvantages of network-based antivirus systems are summarized below:

➢ **Advantages of network-based Antivirus systems:** Advantages of Network based Antivirus systems are:

i. Network-based antivirus systems provide a single barrier behind which all hosts are protected.

ii. A single update of the signature database or scanning algorithms on the network-based antivirus gateway protects all of the systems on either side from viruses flowing in either direction.

iii. Network-based antivirus systems greatly reduce the risk that an unprotected host will be compromised, and mitigate the risk of memory-resident and other viruses that are a challenge for host-based antivirus systems.

iv. Network-based antivirus systems reduce the load on email servers by eliminating infected emails before they reach the servers,

v. Network-based antivirus systems are well positioned in the network to scan Web and other traffic that tends to bypass conventional host-based antivirus systems.

- **Disadvantages of network-based Antivirus systems:** Disadvantages of Network based Antivirus systems are:
    i. For a typical file, many millions of comparisons may be required to determine if it is free from infection.
    ii. Larger signature databases and longer, more complex signatures require more time for an antivirus system to scan and reduce the performance.
    iii. Network-based antivirus systems have to scan packets after reassembling them in buffers that can cause the drop of packets when the buffer is full.
    iv. Network-based antivirus systems typically employ dedicated platforms that have their own vulnerabilities.

In this chapter various types of attacks are studied, a brief study of "how a professional attacker attacks on network" is carried out. Different types of network security tools are discussed with their advantages and disadvantages.

In the next chapter the detail of one of the network security tool called "Intrusion Detection System" is given. The structure of IDS, its functions, IDS classifications and its limitations are discussed.

Chapter 2

Detail of Literature Survey

2.1 Introduction to IDS

This section is the output of the literature survey on Intrusion Detection Systems. The goal is to give details about IDS and their types how IDS works and its architecture. Intrusion Detection can be defined as the act of detecting actions that attempt to compromise the confidentiality, integrity or availability of a resource. More specifically, the goal of intrusion detection is to identify entities attempting to subvert in-place security controls.

2.2 Structure and architecture of IDS: Various components of IDS can be seen from figure 2.1. IDS components [15] are explained as:

- **Data gathering device:** Responsible for collecting data from the monitored system. Act as an agent who continuously watches or monitors the network in real time. Example types of input to a sensor are network packets, log files, and system call traces. Sensors collect and forward this information to the analyzer.
- **Detector:** Processes the data collected from sensors to identify intrusive activities and uses a system of rules to generate alarms from security events received.
- **Knowledge base:** Contains information collected by the sensors but in preprocessed format (e.g. knowledge base of attacks and their signatures, filtered data, data profiles, etc.). This information is usually provided by network and security experts. In database all the information about the attack signatures or pattern should be present that are previously detected. When the sensor detects some kind of malicious activity or signature it matches it with the current database, and report to attack response module.
- **Configuration device:** Provides information about the current state of the Intrusion Detection System.

- **Response component**: Initiates actions when an intrusion is detected. These responses can either be automated or involve human interaction. Based upon the type of configuration the Response Module can either send an alarm or an email notification about the intrusion detected to the administrator

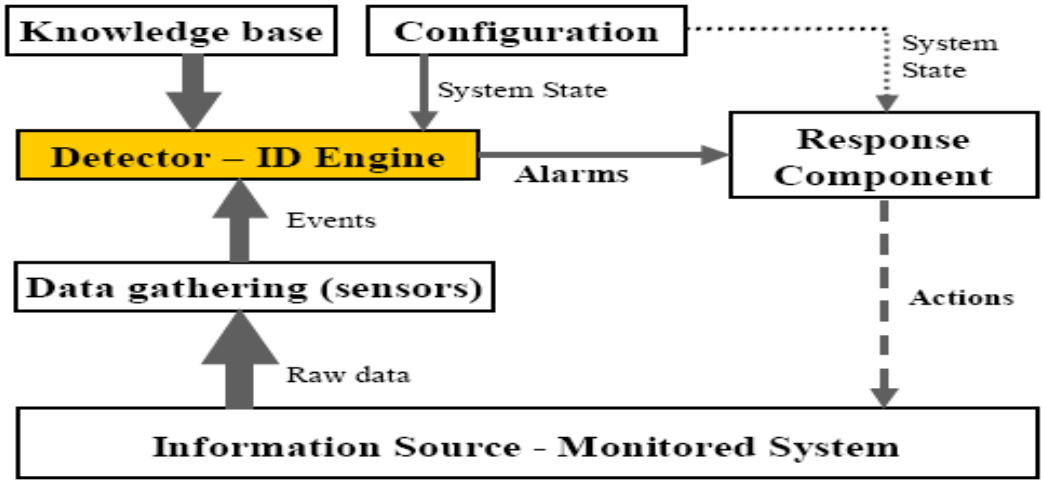


Fig 2.1: IDS Architecture [15]

2.3 Functions of IDS

The purpose of an Intrusion Detection System (IDS) is to detect both external attacks on and internal misuse of computer and network resources or information residing in these resources. The main challenge in intrusion detection is that of separating anomalous events from normal events. Anomalous events can include actual attacks against a computer system or more subtle and hence difficult to detect, probes that are aimed at information reconnaissance. Another challenge in ID is that of false positives. False positives in ID occur when an IDS reports an intrusion as occurring when in fact it has not. It has been argued that it is actually this false alarm rate that is the limiting factor in a IDSs performance. The intent of a network IDS is to guide the analyst towards network-events that are malicious. Following are some of the functions an IDS provides [16]:

- Monitoring and analysis of user and system activity.
- Auditing of system configurations and vulnerabilities.
- Assessing the integrity of critical system and data files.

- Recognition of activity patterns reflecting known attacks.
- Statistical analysis for abnormal activity patterns.
- Operating system audit trail management, with recognition of user activity reflecting policy violations.
- Some systems provide additional features, including: Automatic installation of vendor-provided software patches.
- Installation and operation of decoy servers to record information about intruders.

2.4 Reasons to acquire and use IDS

IDS have gained acceptance as a necessary addition to every organization's security infrastructure. Since they are first put on the security market, those organizations have several compelling reasons to acquire and use IDSs. Some of them are listed below:

- To prevent problematic behaviors by increasing the perceived risk of discovery and punishment for those who would attack or otherwise abuse the system.
- To detect attacks and other security violations that is not prevented by other security measures.
- To detect and deal with the preambles to attacks (commonly experienced as network probes and other reconnaissance activities).
- To document the existing threat to an organization.
- To act as quality control tool for security design and administration, especially for large and complex enterprises.
- To provide useful information about intrusions that take place, allowing detailed analysis, recovery, and correction of causative factors.

2.5 Taxonomy of IDS:

A taxonomy presented below discusses the intrusion detection methods on the basis of six criteria to classify IDS. Figure 2.2 presents the taxonomy of Intrusion Detection System on the bases of six different criteria. Intrusion Detection Systems can be classified on the basis of data source, processing, time of detection, environment, architecture, and reaction which are shown in figure 2.2.

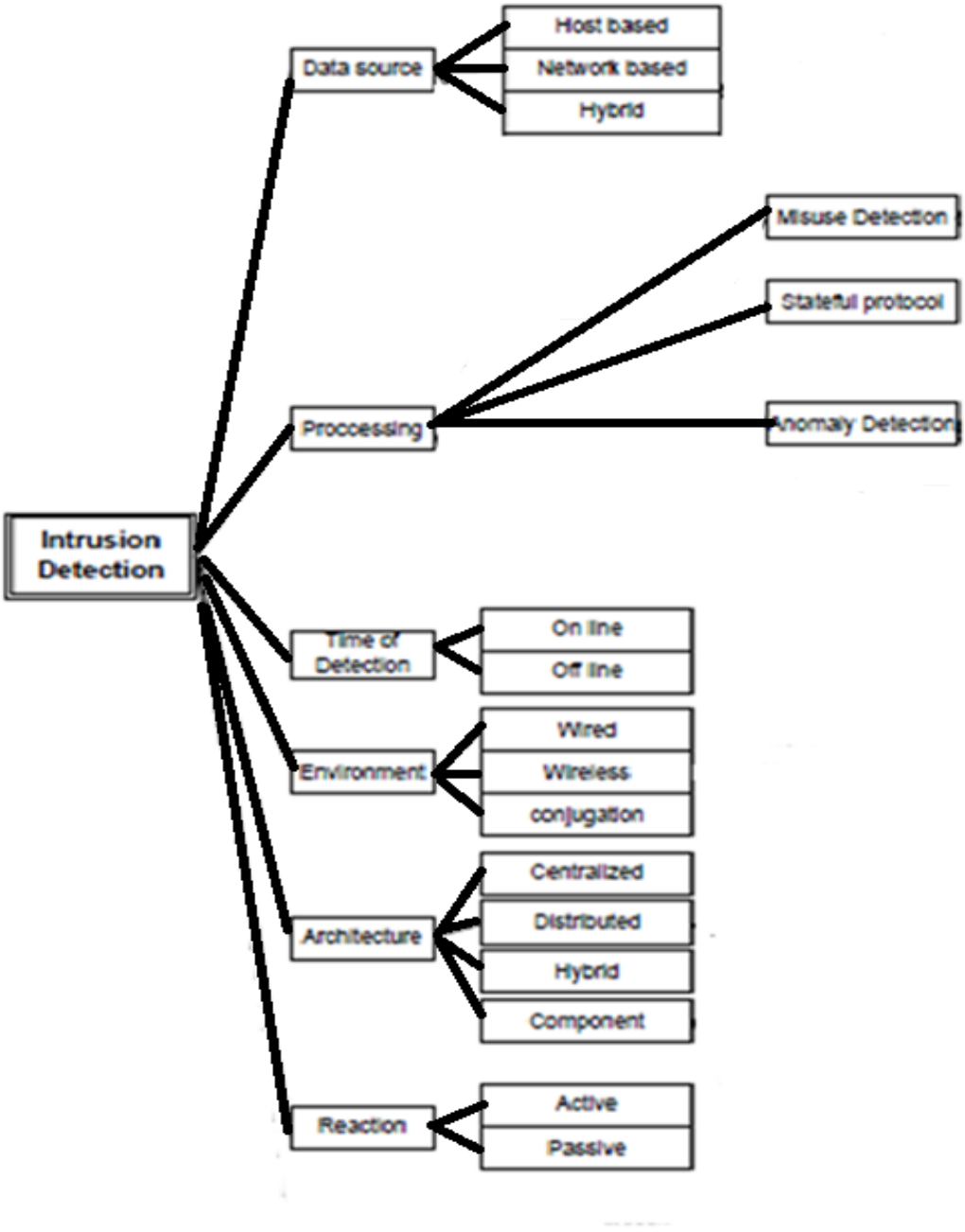


Fig 2.2: Taxonomy of IDS [23]

The criteria for discussing various types of Intrusion Detection Systems are discussed in detail as following:

- **Data source:** IDSs typically perform extensive logging of data that is related to detected events. These data can be used to confirm the validity of alerts, investigate incidents, and correlate events between the IDS and other logging sources. On the basis of data source IDS can be of following types:

i. ***Host-Based IDS:*** HIDS shown in figure 2.3 is an intrusion detection system which monitors the characteristics of a single host and the events occurring within that host, for suspicious activity. Host intrusion detection systems are installed locally on host machines making it a very versatile system compared to NIDS. HIDS can be installed on many different types of machines namely servers, workstations and notebook computers. Examples of HIDS include OSSEC [17] and Tripwire [18].

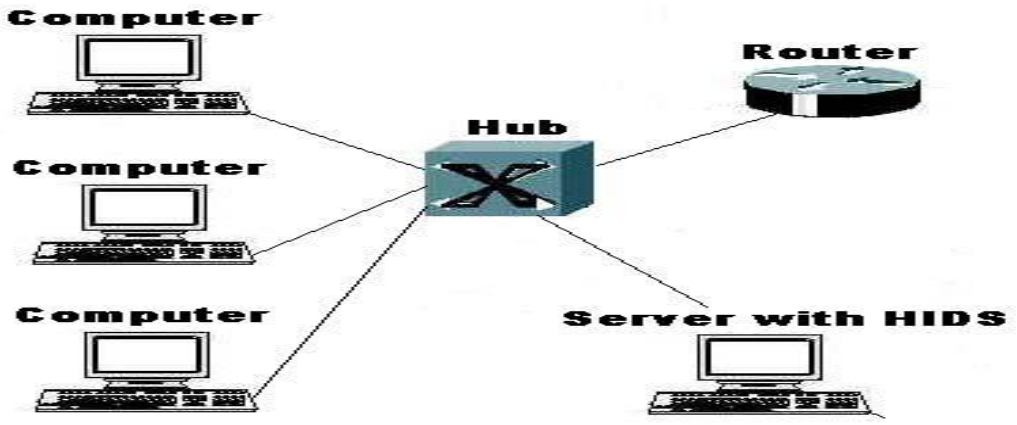


Fig 2.3: HIDS [16]

ii. ***Network-Based IDS:*** NIDS shown in figure 2.4 is an intrusion detection system that applies a network based approach towards an intrusion [19]. In this system we assign a server node for a part of the network. Now if we need to send some data to any node of that specific part of the network, then we send it to the server and then the server redirects the data to its original destination. The server has the system installed in it to detect an intrusion. So, the server detects it and discards the data if it is an intrusion. Otherwise it sends the verified data to the destination node. Example includes Snort [27] and Bro [20]

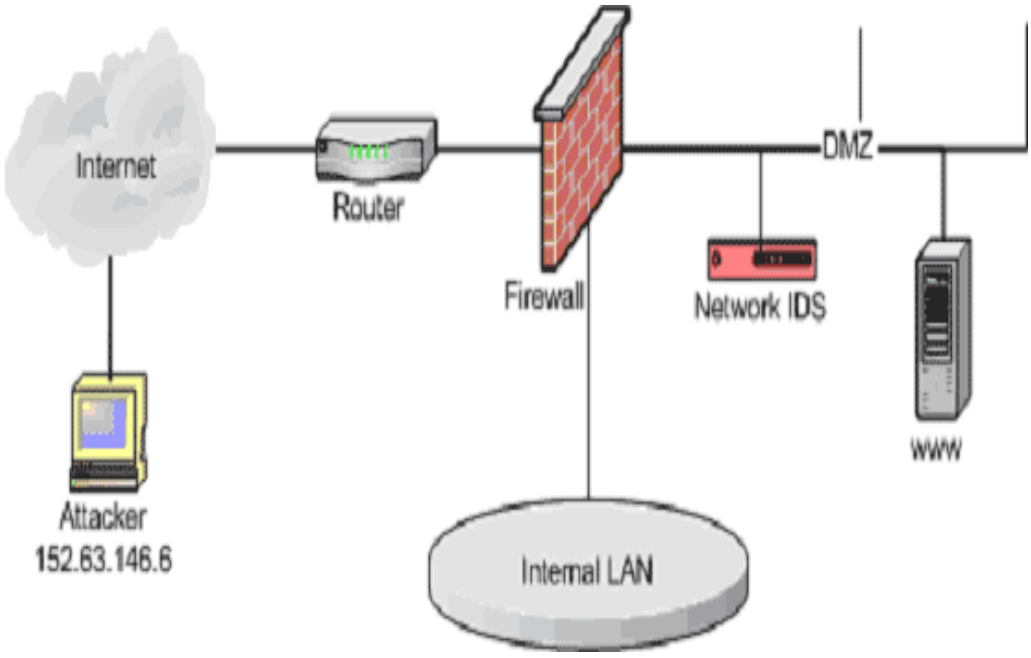


Fig 2.4: NIDS [16]

iii. ***Hybrid:*** Hybrid intrusion detection systems are the systems in which both host based and network based intrusion detection systems can be used simultaneously.

- **Processing:** Once required data collected, an IDS analysis engine processes these data in order to identify intrusive activities. On the basis of processing of data Intrusion Detection Systems are of following types:

i. ***Misuse detection:*** The main object of misuse detection [21] focuses to use an expert system to identify intrusions based on a predetermined knowledge base. Misuse detection can be performed with following methods:

a) Signature based: Matching available signatures in its database with collected data from activities for identifying intrusions.

b) Rule based: Rule based system [22] uses a set of "if-then" implication rules to characterize computer attacks.

c) State transition: In this approach IDSs try to indentify intrusion by using a finite state machine that deduced from network. IDS states correspond to different states of the network and an event make transit in this finite state

machine. An activity identifies intrusion if state transitions in the finite state machine of network reflect to sequel state.

ii. ***Stateful protocol analysis:*** This method compares predetermined profiles of generally accepted definitions of benign protocol activity for each protocol state against observed events to identify deviations.

iii. ***Anomaly detection:*** This method works by using the definition "anomalies are not normal". There are many anomaly detection that proposed algorithms with differences in the information used for analysis and according to methods that are employed to detect deviations from normal behavior. But the most important object is the anomaly detector that must be able to distinguish between the anomaly and normal behavior properly.

   a) Statistical based methods: Statistical methods monitor the user/network behavior by measuring certain variables statistics over time
   b) Distance based methods: These methods try to overcome limitations of statistical outlier detection approach when the data are difficult to estimate in the multidimensional distributions.
   c) Rule based: In rule based systems, IDSs have defined the knowledge of normal behavior of user/network and identified intrusion by comparison this predefined normal behavior with user/network current activities.
   d) Profile based methods: This method is similar to rule based method but in this type, profile of normal behavior is built for different types of network traffics, users, and all devices and deviance from these profiles means intrusion.
   e) Model based methods: Other approaches based on deviance normal and abnormal behavior is modeling them but without creating several profile for them. In model based methods, researchers attempt to model the normal and/or abnormal behaviors and deviation from this model means intrusion.

- **Time:** In time aspect consideration, IDSs have two main groups: online IDS that tries to detect intrusion in real time (or near real-time) as when the live network traffic coming over network and off-line IDS that performs post-analysis data for detecting intrusions means the data is first collected and then analysis is performed.

- **Environment:** Easy accessibility condition in this kind of networks causes their vulnerability against wired networks to high. The level of vulnerability has made it necessary to consider security issues in wireless networks more than before and nowadays new group added to IDS categories for Wireless networks that named WIDS (Wireless Intrusion Detection System). WIDS which monitors wireless network traffics and analyze them to identify suspicious activity involving the wireless networking protocols. Wireless network have two main groups: with infrastructure (e.g. WLAN) and without infrastructure (e.g. Adhoc). Intrusion Detection in wireless networks with infrastructure is similar to regular IDS that exist in wired network. But in wireless network without infrastructure has different situation for performing security issues, because in these types of wireless networks there are specific protocols used in them that challenge regular IDS to work properly in networks with these protocols. In this kind of wireless networks, detecting intrusion differ from wireless networks with infrastructure because tasks perform distributed frequently and necessary to improve regular IDS methods for these types of wireless networks.
- **Architecture:** Most intrusion detection systems are centralized architecture and detect intrusions that occur in a single monitored system/network. But nowadays several attacks appear that have distributed architecture and centralized processors are not able to process collected data from massive network or distributed attacks (e.g.DDoS). In centralized IDS, the analysis of data is performed on a fixed number of locations. But in distributed IDS (DIDS) the analysis of data is performed on a number of locations that is commensurate to number of available systems in network. In wireless network without infrastructure one force to use DIDS because one can't set a fixed location/host for using centralized IDS. Recently, New methods appear in distributed IDS categories with name GIDS (Grid Intrusion Detection system), which uses Grid computing resources to detect intrusion packets.
- **Reaction:** Regularly, IDSs react against attacks passively [24]. IDSs have passive reaction and simply inform administrator of a malicious event, without any countermeasure. In passive reaction, the most important issue is the speed of notification when attacks occur in network [25]. IDSs also beget an active reaction

when attacks occur and response to critical events. But active IDSs generally don't act perfect countermeasure against intrusion because with doing that, IDSs need more process resources and concentrate on detecting and counter measuring capabilities in one system that is not recommended at all.

2.6 Possible results of IDS

Various types of IDS products have the same critical job of being accurate enough to differentiate between the good and bad traffic that gets into the network. The following are all possible results of intrusion detection [26]:

- Undetected bad traffic (false negative)
- Detected bad traffic (true negative)
- Good traffic that the system thinks is bad (false positive – false alarm)
- Good traffic that the system identifies as good (true positive)

- **Undetected bad traffic:** Failure to identify malicious traffic as an attack. This is the worst thing that can happen, because it means the IDS failed to do its job. Failing to detect an attack can occur when an IDS does not have adequate or comprehensive intrusion detection mechanisms in place. It also occurs when new attacks are created and then missed by poorly implemented detection mechanisms .While it is virtually impossible to detect every attack, the goal of any system should be to minimize the number of undetected attacks.
- **Detected bad traffic:** Identifying "real attacks" as an attack. This is the ideal result of IDS. The ability to detect bad traffic with speed and reliability is referred to as intrusion detection accuracy. All other functions of the system hinge on this capability. The more accurate the system, the more you can trust its abilities. A system must have proven accuracy before enabling it to take the necessary actions (such as dropping the connection) to secure the network.
- **Good traffic that the system thinks is bad:** False alarm or false positive. This is the most troublesome and time-consuming aspect of IDS solutions. It occurs when the IDS sees something in legitimate and benign traffic that makes it believe there is an attack. It is detrimental because each and every alarm needs to be investigated in order to determine whether an attack was successful and assess any resulting damage.

Every moment spent investigating a false positive reduces the time available to investigate real threats. The result is that false positives can erode trust in the product; sometimes causing real attack alarms to be overlooked.

- **Good traffic that the system identifies as good:** An ideal result of intrusion detection mechanisms, identifying good traffic for what it is – good traffic

2.7 IDS limitations

Although Intrusion detection system helps in securing our network from various attacks and viruses, but there is certain limitation of these systems as discussed below:

- IDS are not an alternative for a weak identification and authentication mechanisms.
- They are unable to conduct monitoring or to investigate attack without human intervention.
- They are unable to deal with problem with integrity and quality of information provided by the system.
- They are unable to analyze all the traffic on a busy network.
- They cannot always deal with problems involving packet-level attacks.
- False negative occurs when the IDS fail to identify an intrusion when one has in fact occurred.
- False positive occurs when the IDS incorrectly identifies an intrusion when none has occurred.
- IDS is passive and reactive, its main goal is to detect attack actions, not to prevent them [8].
- There are higher misinformation rate, it is difficult to make the detector and access control policy more exact and effective [8].

In this chapter Intrusion Detection System is discussed with its architecture, its functions, classifications and limitations. Now in the next chapter one of the network based Intrusion Detection System called Snort will be discussed in detail.

Chapter 3

Snort NIDS

3.1 Introduction to Snort

Snort [27] is a small, lightweight open source IDS written by Martin Roesch which has become the most widely used IDS. Snort is an open source Intrusion Detection System, which may also be configured as an Intrusion Prevention system [28] for monitoring and prevention of security attacks on networks. It is a cross-platform, lightweight network intrusion detection tool that can be deployed to monitor small TCP/IP networks and detect a wide variety of suspicious network traffic as well as outright attack. Snort performs protocol analysis, and content searching/matching, it is commonly used to actively block or passively detect a variety of attacks and probes, such as buffer overflows, stealth port scans, web application attacks, SMB probes, and OS fingerprinting attempts, amongst other features.

3.2 Snort components

Snort is developed with the performance, simplicity, and flexibility in mind. Snort is logically divided into multiple components. These components work together to detect various attacks and to generate output in a required format. These components ride on top of the Libpcap or WinPcap promiscuous packet capturing library, which provides a portable packet sniffing and filtering capability. Snort consists of the following major components [29] shown in figure 3.1.

- **Packet capture block:** This module is used for capturing the packets and passing the packets to other modules. It uses Libpcap [30], Winpcap [31] or iptables [32].
- **Decoder:** It is the module responsible to perform the syntax analysis at layer 2, 3 and 4 of the IP packet (MAC, IP and TCP/UDP layer).
- **Preprocessors:** It is the block where multiple preprocessors can be loaded at boot time to analyze protocols of layers above the TCP/UDP one with custom made C/C++ programs.

- **Detection engine:** It is the block where signatures and rules are checked to analyze protocols of layers above the TCP/UDP one.
- **Logging & Alerting module:** It is the block managing the log output. The output log is configurable depending on user needs. Alerting based on the rules are also generated by this component.

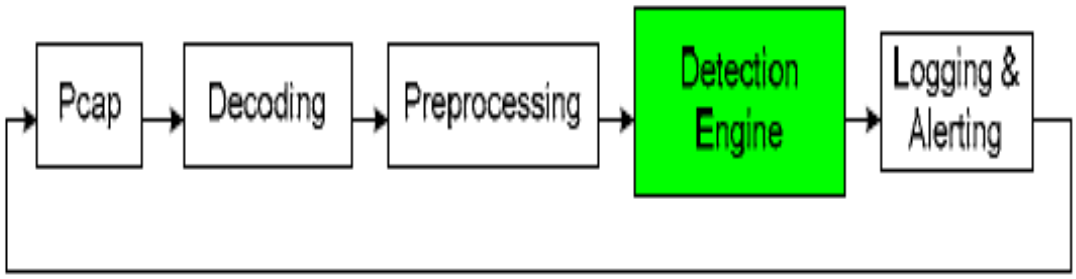


Fig 3.1: Snort Components [33]

The components along with their function and description are shown in Table 3.1 below.

Component Name	**Description**
Packet Capture block	Captures packets using Libpcap or Winpcap.
Decoder	Prepares packet before sending to detection engine.
Preprocessors	Arrange or modify data packets, can rearrange packets if needed.
Detection Engine	Detect any activity against the rules.
Logging & Alerting Module	Provides the output

Table 3.1: Snort Components [29]

3.3 Snort key features

The basic features of snort includes packet capturing, logging, performing real time

analysis, content searching and matching and detecting attacks which are explained in detail as:

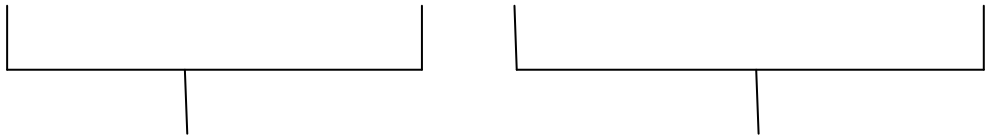
- **Packet capturing and logging:** Snort captures the packets using Libpcap or Winpcap to capture the packets coming on an interface.
- **Real time traffic analysis:** Snort is capable of performing the analysis on real time packets on the network. Although it is also capable of performing analysis on captured packets which can be captured using various tools like Wireshark.
- **Content searching/matching:** Snort performs the content searching of the packet's content against the rules in the rules file.
- **Block or passively detect a variety of attacks and probes:** Snort can block the packets over the network when it is operating in inline mode. It is able to detect malicious activities over the network.

3.4 Writing snort rules

Snort uses a simple, lightweight rules description language that is flexible and quite powerful. There are a number of simple guidelines to remember when developing Snort rules. Snort rules have the following parts:

- Rule Header
- Rule options

Alert tcp $HOME_NET any -> any 80 (msg: "GET Match"; content: "|47 45 54|" ;)



Rule Header Rule options

3.4.1 Rule header

The rule header [34] contains the information that defines the who, where, and what part of a packet, as well as what to do in the event that a packet with all the attributes indicated in the rule should show up. Rule header in a Snort contains all those portions which are the compulsory part. It includes following parts:

- **Rule Actions:** The rule action tells Snort what to do when it finds a packet that matches the rule criteria. There are 5 available default actions in Snort:
    i. alert: Generate an alert using the selected alert method, and then log the packet
    ii. log: Log the packet
    iii. pass: Ignore the packet
    iv. activate: Alert and then turn on another dynamic rule
    v. dynamic: Remain idle until activated by an activate rule, then act as a log rule
    vi. drop: Make iptables drop the packet and log the packet
    vii. reject: Make iptables drop the packet, log it, and then send a TCP reset if the protocol is TCP or an ICMP port unreachable message if the protocol is UDP.
    viii. sdrop: Make iptables drop the packet but does not log it.
- **Protocol:** The next field in a rule is the protocol. There are four protocols that Snort currently analyzes for suspicious behavior– TCP, UDP, ICMP, and IP.
- **IP Addresses:** The keyword any may be used to define any address. The addresses are formed by a straight numeric IP address and a CIDR block. The CIDR block indicates the netmask that should be applied to the rule's address and any incoming packets that are tested against the rule. Source IP and Destination IP are written in the rule header.
- **Port Numbers:** Port numbers may be specified in a number of ways, including any ports, static port definitions, ranges, and by negation. Any ports are a wildcard value, meaning literally any port. Static ports are indicated by a single port number, such as 111 for portmapper, 23 for telnet, or 80 for http, etc. Port ranges are indicated with the range operator:. Port negation is indicated by using the negation operator (!). Both source and destination ports are required in rule. The keyword any may be required to use any port number.
- **The Direction operator:** The direction operator -> indicates the orientation, or direction, of the traffic that the rule applies to. The IP address and port numbers on the left side of the direction operator is considered to be the traffic coming

from the source host, and the address and port information on the right side of the operator is the destination host. There is also a bidirectional operator, which is indicated with a <> symbol. This tells Snort to consider the address/port pairs in either the source or destination orientation.

3.4.2 Rule options:

Rule options [34] form the heart of Snort's intrusion detection engine, combining ease of use with power and flexibility. All Snort rule options are separated from each other using the semicolon (;) character. Rule option keywords are separated from their arguments with a colon (:) character. There are four major categories of rule options.

- General rule options
- Payload detection rule options
- Non-payload detection rule options
- Post-detection rule options

- **General rule options:** These options provide information about the rule but do not have any affect during detection**.** Table 3.2 shows the attributes in general rule options:

Attribute	**Description**
msg	Tells the logging and alerting engine the message to print along with a packet dump or to an alert.
reference	Allows rules to include references to external attack identification systems.
gid	Used to identify what part of Snort generates the event when a particular rule fires.
sid	The sid keyword is used to uniquely identify Snort rules
rev	Used to uniquely identify revisions of Snort rules. Revisions, along with Snort rule id's, allow signatures and descriptions to be defined and replaced with updated information.

classtype	Used to categorize a rule as detecting an attack that is part of a more general type of attack class.
priority	The priority tag assigns a severity level to rules.
metadata	The metadata tag allows a rule writer to embed additional information about the rule, typically in a key-value format.

Table 3.2: General rule options [34]

- **Payload detection rule options:** These options all look for data inside the packet payload and can be inter-related. Table 3.3 shows the attributes in payload detection rule options.

Attribute	**Description**
content	Allows the user to set rules that search for specific content in the packet payload and trigger response based on that data.
rawbytes	Allows rules to look at the raw packet data, ignoring any decoding that was done by preprocessors.
depth	Allows the rule writer to specify how far into a packet Snort should search for the specified pattern.
offset	Allows the rule writer to specify where to start searching for a pattern within a packet.
distance	Allows the rule writer to specify how far into a packet Snort should ignore before starting to search for the specified pattern relative to the end of the previous pattern match.
within	It is a content modifier that makes sure that at most N bytes are between pattern matches using the content keyword.

uricontent	It searches the normalized request URI field.
isdaat	Verifies that the payload has data at a specified location.
pcre	Allows rules to be written using perl compatible regular expressions.
byte test	Tests a byte field against a specific value (with operator).
bytejump	Allows rules to read the length of a portion of data, then skip that far forward in the packet.
ftpbounce	Detects FTP bounce attacks.
asn1	Decodes a packet or a portion of a packet, and looks for various malicious encodings.
cvs	Detects invalid entry strings.

Table 3.3: Payload detection rule options [34]

- **Non-Payload detection rule options:** These options look for non-payload data. Table 3.4 shows the attributes in non-payload detection rule options.

Attribute	**Values**
fragoffset	Allows one to compare the IP fragment offset field against a decimal value.
ttl	Used to check the IP time-to-live value.
tos	Used to check the IP TOS field for a specific value.
id	Used to check the IP ID field for a specific value.
ipopts	Used to check if a specific IP option is present.
fragbits	Used to check if fragmentation and reserved bits are set in the

	IP header.
dsize	Used to test the packet payload size.
flags	Used to check if specific TCP flag bits are present.
flow	Allows rules to only apply to certain directions of the traffic flow.
flowbits	Allows rules to track states during a transport protocol session.
seq	Used to check for a specific TCP sequence number.
ack	Used to check for a specific TCP acknowledge number.
window	Used to check for a specific TCP window size.
itype	Used to check for a specific ICMP type value.
icode	Used to check for a specific ICMP code value.
icmp id	Used to check for a specific ICMP ID value.
icmp seq	Used to check for a specific ICMP sequence value.
rpc	Used to check for a RPC application, version, and procedure numbers in SUNRPC CALL requests.
ip proto	Allows checks against the IP protocol header.
sameip	Allows rules to check if the source ip is the same as the destination IP.
logto	The logto keyword tells Snort to log all packets that trigger this rule to a special output log file.

Table 3.4: Non-payload detection rule options [34]

- **Post-detection rule options:** These options are rule specific triggers that happen after a rule has "fired". Table 3.5 shows the attributes in post- detection rule options.

Attribute	**Description**

session	Extract user data from TCP Sessions.
resp	Attempt to close sessions when an alert is triggered.
react	Implements ability for users to react to traffic that matches a Snort rule by closing connection and sending a notice.
aag	Allow rules to log more than just the single packet that triggered the rule.
activates	Allows the rule writer to specify a rule to add when a specific network event occurs.
activated by	Allows the rule writer to dynamically enable a rule when a specific activate rule is triggered.
Count	It must be used in combination with the activated by keyword. It allows the rule writer to specify how many packets to leave the rule enabled for after it is activated.
replace	Replace the prior matching content with the given string of the same length. Available in inline mode only.
detection filter	Track by source or destination IP address and if the rule otherwise matches more than the configured rate it will fire.

Table 3.5: Post-detection rule options [34]

3.5 Snort rules classification

The rules in Snort are able to perform the analysis on network data on different criteria. The multi-rule search engine is broken into three distinct searches based on unique Snort rule properties:

- Protocol field search
- Generic content search
- Packet anomaly search

i. **Protocol field search:** The protocol field search allows a rule to specify a value in a particular protocol field to search for.

alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg: "TCP packets are detected on our network"; sid: 1000003 ;)

The rule looks for any tcp packet on the network and display the message "*packets are detected on our network*" whenever finds any tcp packet over network.

ii. **Generic content search**: The generic content search allows a rule to specify a byte string to match against anywhere in the packet including the payload data.

alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-IIS cmd.exe access"; flags: A+; content:"cmd.exe"; nocase; classtype:web-application-attack; sid:1002; rev:2;)

The rule [35] looks for the "NIMDA" worm [36] over network and displays the message "WEB-IIS cmd.exe access" if the worm is detected over network.

iii. **Packet anomaly search:** The packet anomaly search allows a rule to specify characteristics of a packet or packet header that is cause for alarm. Many attacks use buffer overflow vulnerabilities by sending large size packets. Using the 'dsize' keyword in the rule, one can find out if a packet contains data of a length larger than, smaller than, or equal to a certain number.

alert ip any any -> 192.168.1.0/24 any (dsize: > 6000; msg: "Large size IP packet detected";)

The rule [35] generates an alert if the data size of an IP packet is larger than 6000 bytes.

3.6 Operation modes of snort: Snort runs in different operational modes which are as follows [35]:

- **Sniffer Mode:** Simply reads the packet off the network and displays them in a continuous stream on the console.

  Command: ./snort –[option]

  Options to run in a sniffer mode are:

  i. v: The command prints the TCP/IP header on the screen.
  ii. vd: The command prints the application data too.
  iii. vde: The command prints the data link layer contents as well.

- **Packet Logger Mode:** Logs the packets to the disk. Options to run snort in sniffer mode are:
    i. ./snort –dev –l ./log : The command logs the packets to the directory specified).
    ii. ./snort –l ./log –b : The command performs the binary log, binary file may be read back using –r switch).
- **Network Intrusion Detection Mode (NIDS):** Most complex and configurable mode. It allows Snort to analyze network traffic against a user defined rule set and performs actions based on detections.

  Command: ./snort – [option] –c snort.conf :

  There are number of ways to configure snort output in NIDS mode. There are total 7 alert modes out of which six modes can be used with -A option. Options to run snort in NIDS mode are shown in table 3.6.

Option	**Description**
-A fast	Writes the alert in a simple format with a timestamp, alert message, source and destination IPs/ports.
-A full	This is the default alert mode and will be used automatically if you do not specify a mode.
-A unsock	Sends alerts to a UNIX socket that another program can listen on.
-A none	Turns of alerting.
-A console	Sends “fast-style” alerts to the console
-A cmg	Generates “cmg style” alerts.

Table 3.6: Configure Snort outputs in NIDS mode [34]

- **Inline Mode:** Obtains packets from Iptables instead of the libpcap or Winpcap and then causes iptables to drop or pass packets based on snort rules that use inline-specific rule types. (drop, reject, sdrop options). This is the mode used when Snort is to act as an IPS.

Table 3.7 showing different operation modes of Snort:

Mode	**Description**
Sniffer Mode	Simply reads the packet and displays them in a continuous stream on the console.
Packet Logger Mode	Logs the packets to the disk.
NIDS Mode	It allows Snort to analyze network traffic against a user defined rule set and performs actions
Inline Mode	Act as intrusion prevention and performs dropping or rejecting the packet.

Table 3.7: Snort operation modes [35]

3.7 A Brief study of Snort

Table 3.8 shows all the information about Snort in short form.

Attribute	**Value**
System name	Snort
Type	NIDS/NIPS (Network intrusion detection/Prevention systems)
IDS/IPS	IDS and IPS Both(IPS in Inline Mode)
Approach	Mainly Signature based but also able to check for anomalies.

License	Open Source
Sniffer version released in	1998
IDS version release in	2003
Engine	Uses human readable flexible Pearl Compatible Regular Expression (PCRE) engine
Developer	Martin Roesch
Source	http://www.snort.org
Platforms	Linux, FreeBSD and Windows.
Standard	De-facto
Packet analysis	Real time, on captured packets.
Modes	• Sniffer Mode • Packet logger Mode • NIDS Mode • Inline Mode
Output formats	• Binary • Libpcap • ASCII [37]
Threats handled	• buffer overflows • stealth port scans • CGI scans • SMB probes • NetBIOS queries • DDoS attacks [38] • Trojan horse attacks • Other types of viruses and worms.

Key features	• Packet capturing and logging • Real time traffic analysis • Content searching/matching • Block or passively detect a variety of attacks and probes.
Advantages	• Light weight intrusion detection • Portability and flexibility • Human readable format logging • Different output modes support • Works also as Tcpdump [39] • Simple rule creation • Efficient rule matching • Open source • GUI support available
Disadvantages	• Work for few protocols only • High packet missing rate compared to others. • Performance decrease as number of rules increase. • Demands more CPU time.

Table 3.8: Snort Brief Study.

Chapter 4

Problem Statement

With the increasing demand of network and advances in the field of network, now a days each organization wants to have their own network and furthermore they want to connect or interact with each other in a reliable way. So network security is becoming more and more important and also getting more complicated issue with recent advances and with increasing demand.

To protect the enterprise, security managers have deployed a variety of technologies including honeypots, firewalls, and intrusion detection/prevention systems. In the thesis the topic under concern is Intrusion Detection System. Intrusion Detection Systems is a device or software application that monitors network and/or system activities for malicious activities or policy violations and produces reports to a Management Station. One of the examples of an intrusion detection system is Snort. Snort is a small, lightweight open source IDS which has become the most widely used IDS. Snort can also be configured as an Intrusion Prevention system.

The objectives of the thesis includes following:

- Our objective is to study and explore a network based intrusion detection system Snort. To perform detailed analysis on working of Snort on two different platforms.
- Install it on both the platforms i.e. on Ubuntu which is a Linux based platform and on Windows7.
- Live traffic will be captured and based upon our own set of signatures we will explore the tool in that traffic to detect intrusions in a network. To create rules in Snort based upon the following criteria:
    i. Based on protocol.
    ii. Based on content.
    iii. Based on port number.

- To run Snort in all the modes as an Intrusion Detection System i.e.
    i. Sniffer mode
    ii. Packet logger mode
    iii. NIDS mode
- One of the main objectives is to create a graphical user interface for Snort rule creation and execution which will let it easy to make the rules in snort and execute them in a simple and easy way. The purpose of graphical user interface is to avoid learning the typical commands to run Snort and rule making process.

Chapter 5

Implementation and Experimental Results

In this chapter configuration steps of Snort on Ubuntu and windows7 are explained. Configuration steps of Mysql as a database in case of Snort are also explained.

5.1 Steps performed during configuration of Snort on Ubuntu

Configuration of Snort on Ubuntu involves the following steps:

- To establish a segregate network using virtualization. VMware Server Console is used to establish a segregate network and Ubuntu 10.10 operating system is installed on it.
- Configuring Snort in Ubuntu 10.10 platform requires some packages. All the dependant packages are grabbed from Synaptic i.e. System > Administration >Synaptic Package .Manager Searches for the following packages were done if not present were installed.

i. Libpcap0.8-dev
ii. libmysqlclient15-dev
iii. mysql-client-5.0
iv. mysql-server-5.0
v. bison
vi. flex
vii. apache2
viii. libapache2-mod-php5
ix. php5-gd
x. php5-mysql
xi. libphp-adodb
xii. php-pear
xiii. libc6-dev
xiv. g++
xv. gcc

- Snort was downloaded under root privileges from www.snort.org and installed by Tar command “tar -xvzf /root/desktop/snort-2.7.0.tar.gz”.
- Configuration of Mysql with snort is done as follows:

i. Changes to configuration file of Snort is done which is stored in /etc/snort /snort.conf. "VAR HOME_NET any" is changed to "VAR HOME_NET 192.168.84.128/24”.

ii. "VAR EXTERNAL_NET any" is changed to "VAR EXTERNAL_NET !$ HOME_NET".

iii. "VAR RULE_PATH /rules" is changed to "VAR RULE_PATH /etc/snort/rules"

iv. "# Output database: log, Mysql, user=" "#" is removed from the front of this line. Leave the "user=root", "password=password" is changed to ″password=new _root_ password", "dbname=snort".

v. Now we need to set up Mysql server log to Mysql server with # Mysql -u root –p.User name and password is entered. After logging you Mysql> password is set for root@localhost=password=”new_root_password”.

vi. Database snort is created in Mysql with “*Create Database*” snort command and exit. Inbuilt database schema of Snort is extract to the Mysql database with command “# Mysql -D snort -u root -p < /root/snort-2.8.0/schemas/create_mysql” where snort-2.8.0 is the version of snort installed it can be different depending upon which version to installed.

Now the Snort is ready to use and perform Experiments.

5.2 Steps performed during configuration of Snort on window7.

Installing the base Snort system requires two components: the WinPcap packet capture library, and the Snort IDS program itself. In the following sections we configure and install both WinPcap and Snort. For logging purpose Mysql database has not been used. So the window installation does not considering those steps.

- **Winpcap:** To configure Winpcap do the following:

i. Download the latest installation file from *http://winpcap.polito.it/install/default.htm*

ii. Double-click the executable installation file and follow the prompts. WinPcap installs itself where it belongs.

- **Snort:**

  Snort.org distributes a convenient install package for Windows available at its Web site: *http://www.snort.org/dl/binaries/win32/*

i. Download this package and perform the following steps to install Snort: Double-click the executable installation file. The GNU Public License appears.

ii. Click the I Agree button. Installation Options window appears.

iii. In the Installation Options dialog box, click the appropriate boxes to select from among these options.

iv. Click the Next button. The Choose Components window appears.

v. In the Choose Components window, select the components you want to install and then click the Next button. The Install Location window appears.

vi. Choose a directory to install to.

vii. Click the Install button.

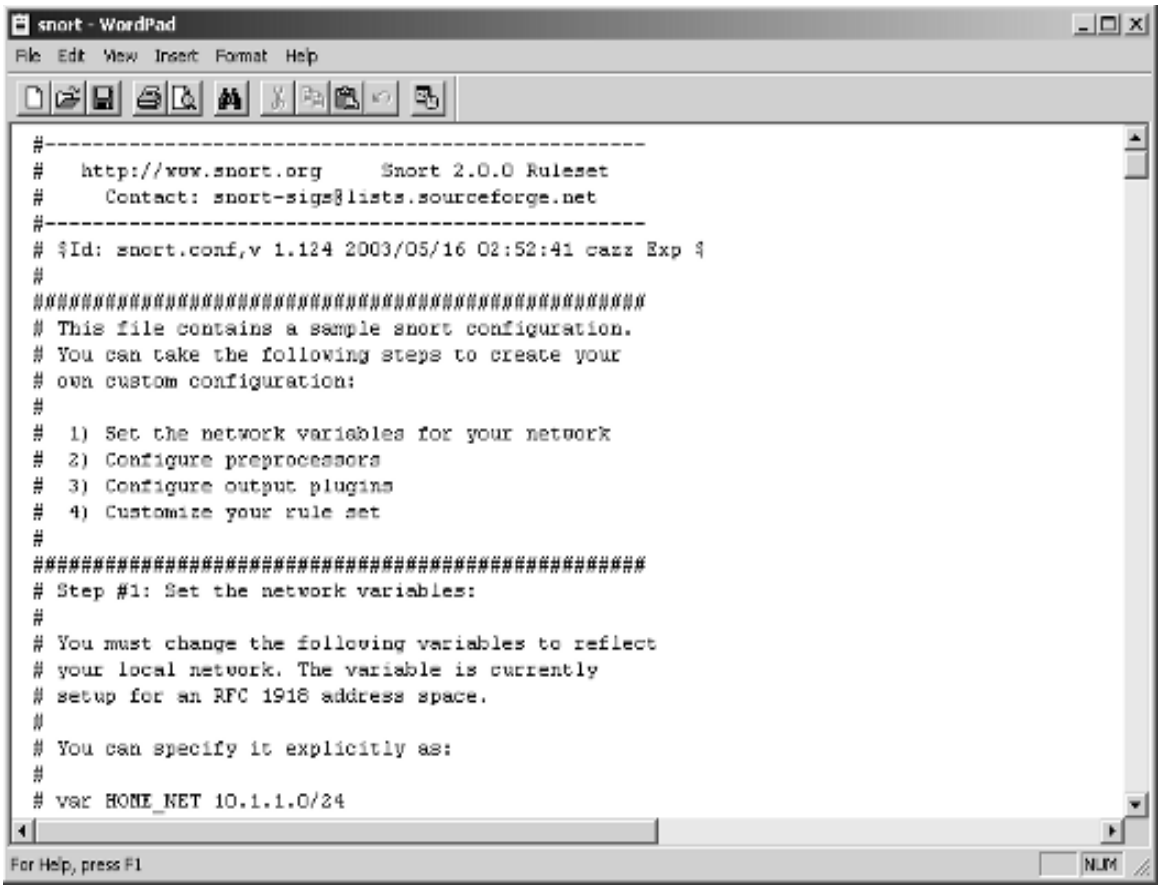
viii. When the installation is complete, click the Close button. An information window appears.

ix. Click the OK button.

You're done! Now it's time to move on to configuring your Snort system. A new Snort installation requires a few configuration points. Conveniently, one file has all the configuration settings required (*Snortpath* is the path to your Snort installation):

Snortpath/etc/snort.conf

When you're ready to configure Snort, open snort.conf in a text editor which is shown as in Snapshot 5.1.



Snapshot 5.1: Snort.conf file

The following configuration options in the snort.conf file are essential to a properly functioning Snort installation:

- **Network settings:** The network settings allow you to set Snort to monitor any range of network IP addresses, from a single IP address, several IP addresses in groups or individually, and entire IP subnets. You can configure the IP address range and the subnet.

  You can control the network range that Snort monitors by changing the var HOME_NET setting in snort.conf. Your options are:

i. Entire network: By default, snort.conf contains the following line, which monitors the entire

   Local network:

   var HOME_NET any

ii. Single IP address: To monitor a single IP or computer insert the IP address range and the subnet of the network or host into snort.conf. To do this, replace the existing

var HOME_NET configuration line with this form:

var HOME_NET IPAddressRange/Subnet

iii. Multiple hosts: One can specify a number of hosts within the network space you are monitoring by listing them in the var HOME_NET configuration statement. The line takes this form:

var HOME_NET IPAddressRange/Subnet,IPAddressRange/Subnet.

- **Rules settings:** So Snort can detect attacks and alert you when attacks occur, Snort needs to know where its rulebase is (and you need to know it if you want to write new rules). To set the rules path in the snort.conf file, replace the existing var RULE_PATH line with this form (*Snortpath* is the location of the Snort install): var RULE_PATH *SnortPath*\rules
- **Output settings:** Output settings are very important in Snort, for they define how Snort's information will be presented to you.

Alert output

The alert output setting is added to the snort.conf configuration file. The snort.conf file will also come in handy when we port that information into our Mysql database. Follow these steps:

i. Find the output line that appears by default as:

output log_tcpdump: tcpdump.log

Because the default line begins with the comment character (#), Snort ignores it.

ii. Change the preceding default output line to this:

output alert_fast: alert.ids

This setting creates a flat text file in the 'log' directory where Snort appends each alert created when one of its rules fires on incoming network packets.

- **Include settings:** Two standard Snort configuration files must be referenced for Snort to properly classify and provide references to the alerts it generates: classification.config and reference.config.

i. classification.config: classification.config holds alert levels for the rules that Snort monitors against network traffic. To set the classification.config file in the snort.conf configuration file, follow these steps:

   a) Find this default line in the snort.conf file:
      Include classification.config

   b) Insert the actual path for the classification.config file into the preceding Include line, like this:
      Include *SnortPath*\etc\classification.config
      For example, the actual snort.conf file on our test system has this line:
      Include C:\Snort\etc\classification.config

ii. reference.config: reference.config contains URLs referenced in the rules that provide more information about the alert event. To set the reference.config file in the snort.conf file, follow these steps:

   a) Find this default line in the snort.conf file:
      Include reference.config

   b) Insert the actual path for the reference.config file into the preceding Include line, like this:
      Include *SnortPath*\etc\reference.config

Now Snort is ready to work. Experimentation can be performed on Snort.

5.3 Milestones covered and Experimentation results

The section covers the milestones covered during entire thesis. Milestones are divided into two parts. Some of the experimentation work will be shown using Ubuntu and rest of the work on windows 7.

5.3.1 Milestones covered in Ubuntu

Various experiments performed on Ubuntu include the following.

- **Protocol, Content and Port based detection capabilities of Snort with a rule driven language.**

  Snort is capable of signature, Protocol & Anomaly detection as well. At the heart of all these detection lie the detection engine and the rules for identifying

malicious traffic. The rule set of Snort is very powerful, flexible, and it is easy to understand and construct new rules.

i. Example of protocol detection rule:

Snort is provided with rule set for detecting protocols and suspicious behavior of these protocols in use. Snort rules have the protocol field that allows the detection of the protocol.

alert tcp $HOME_NET any -> $EXTERNAL_NET any(msg:"TCP packets detected"; sid:10000017;)

Run snort using: *snort –c snort.conf –A console –i eth0*

Snapshot 5.2 shows the alerts when TCP packets are detected.

```
        Using PCRE version: 8.02 2010-03-19

        Rules Engine: SF_SNORT_DETECTION_ENGINE  Version 1.11  <Build 17>
        Preprocessor Object: SF_SSLPP  Version 1.1  <Build 3>
        Preprocessor Object: SF_FTPTELNET  Version 1.2  <Build 12>
        Preprocessor Object: SF_DNS  Version 1.1  <Build 3>
        Preprocessor Object: SF_SMTP  Version 1.1  <Build 8>
        Preprocessor Object: SF_Dynamic_Example_Preprocessor  Version 1.0  <Build 1>
        Preprocessor Object: SF_DCERPC2  Version 1.0  <Build 2>
        Preprocessor Object: SF_SSH  Version 1.1  <Build 2>
        Preprocessor Object: SF_DCERPC  Version 1.1  <Build 5>
Not Using PCAP_FRAMES
06/09-08:33:51.862166  [**] [1:10000017:0] TCP packets detected [**] [Priority: 0] {TCP} 192.168.1
.27:32931 -> 74.125.236.31:80
06/09-08:33:51.920385  [**] [1:10000017:0] TCP packets detected [**] [Priority: 0] {TCP} 74.125.23
5.31:80 -> 192.168.1.27:32931
06/09-08:33:52.033736  [**] [1:10000017:0] TCP packets detected [**] [Priority: 0] {TCP} 74.125.23
5.31:80 -> 192.168.1.27:32931
06/09-08:33:52.033770  [**] [1:10000017:0] TCP packets detected [**] [Priority: 0] {TCP} 192.168.1
.27:32931 -> 74.125.236.31:80
06/09-08:33:52.677209  [**] [1:10000017:0] TCP packets detected [**] [Priority: 0] {TCP} 74.125.23
5.31:80 -> 192.168.1.27:32931
06/09-08:33:52.677242  [**] [1:10000017:0] TCP packets detected [**] [Priority: 0] {TCP} 192.168.1
.27:32931 -> 74.125.236.31:80
06/09-08:33:52.728608  [**] [1:10000017:0] TCP packets detected [**] [Priority: 0] {TCP} 74.125.23
5.31:80 -> 192.168.1.27:32931
06/09-08:33:52.728641  [**] [1:10000017:0] TCP packets detected [**] [Priority: 0] {TCP} 192.168.1
.27:32931 -> 74.125.236.31:80
06/09-08:33:52.780523  [**] [1:10000017:0] TCP packets detected [**] [Priority: 0] {TCP} 74.125.23
```

Snapshot 5.2: Alerts for tcp packets

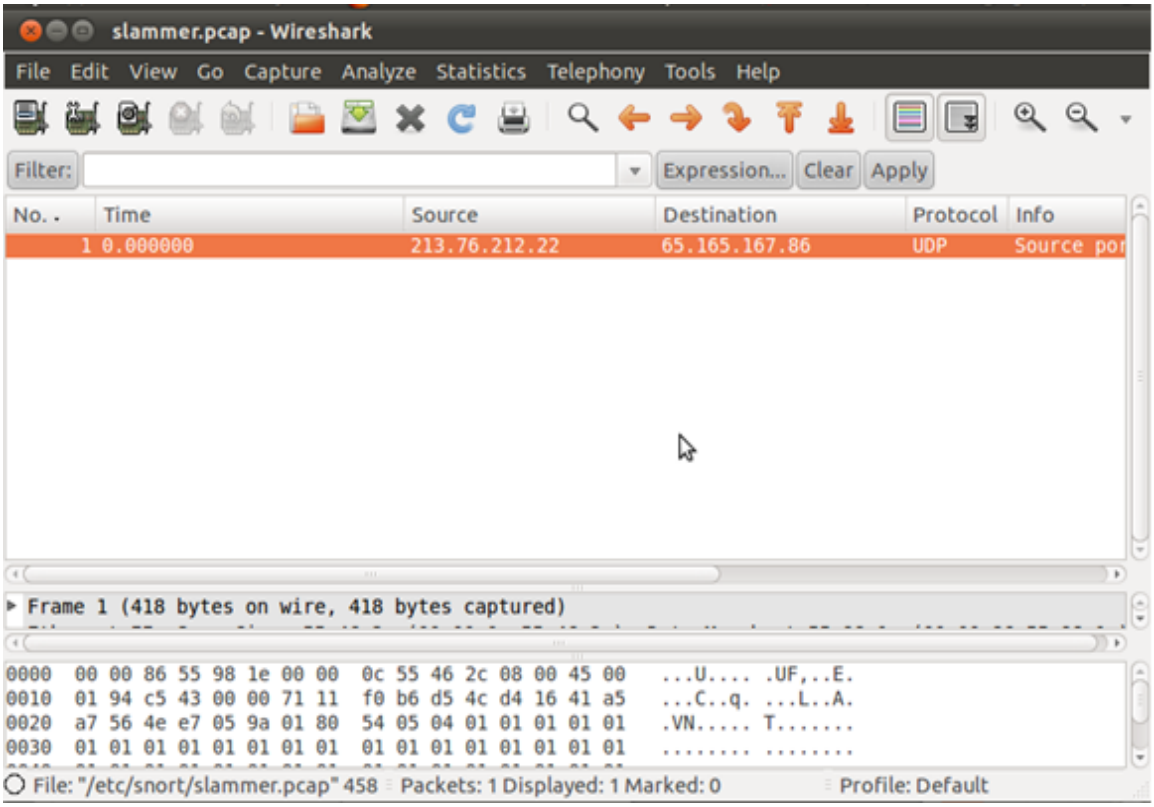
ii. Example of content based detection rule:

Snort handles its signatures based detection with the rules created for it. In the second half of 2001 new and powerful worms were found on the Net, such as Code Red, Code Red II and Nimda, sql slammer worm. Example shows the detection of slammer worm [40] over the network.

alert udp any any -> any any (content:"|04 01 01 01 01 01 01 01|";

*msg:"**slammer worm detected **"; sid: 10000012;)*

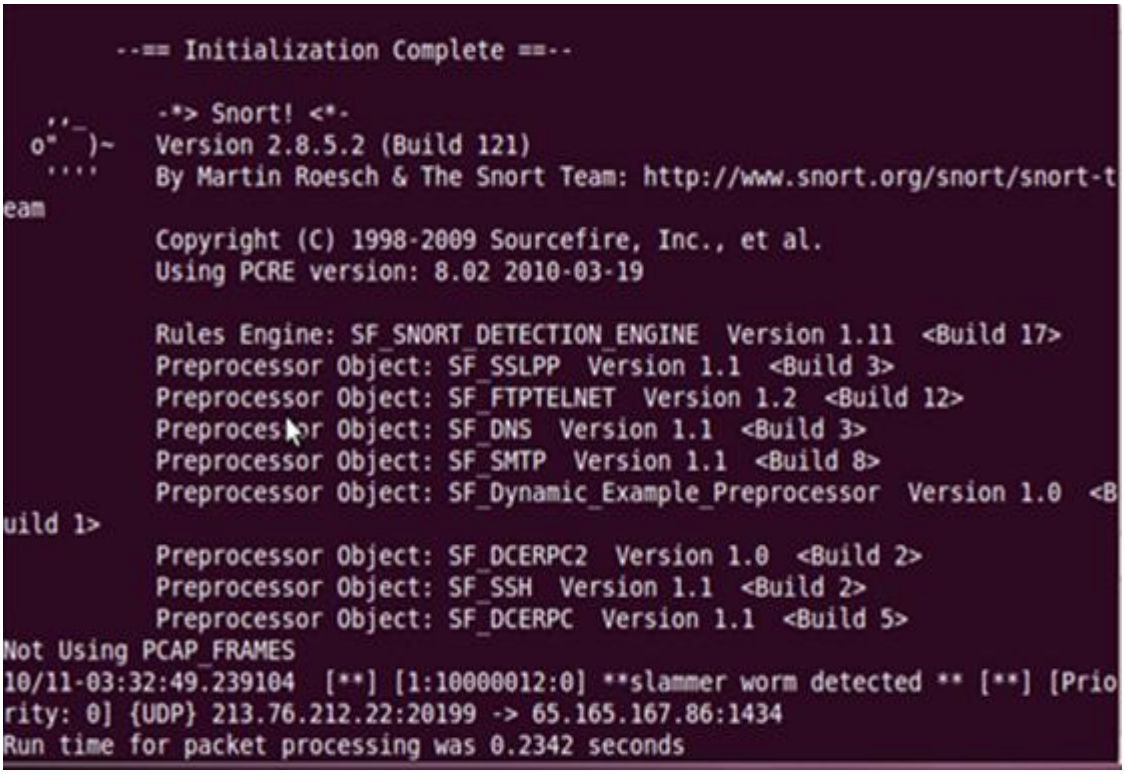
Snapshot 5.3 shows the packets which contain slammer worm opened using Wireshark.



Snapshot 5.3: Packet containing slammer worm

Run Snort using: *Snort –A console –c snort.conf –r slammer.pcap*

Where slammer.pcap is a file which contains captures worm. The alerts for slammer worm result are shown in the Snapshot 5.4.



Snapshot 5.4: Alerts for slammer worm

iii. Example of port based detection rule.

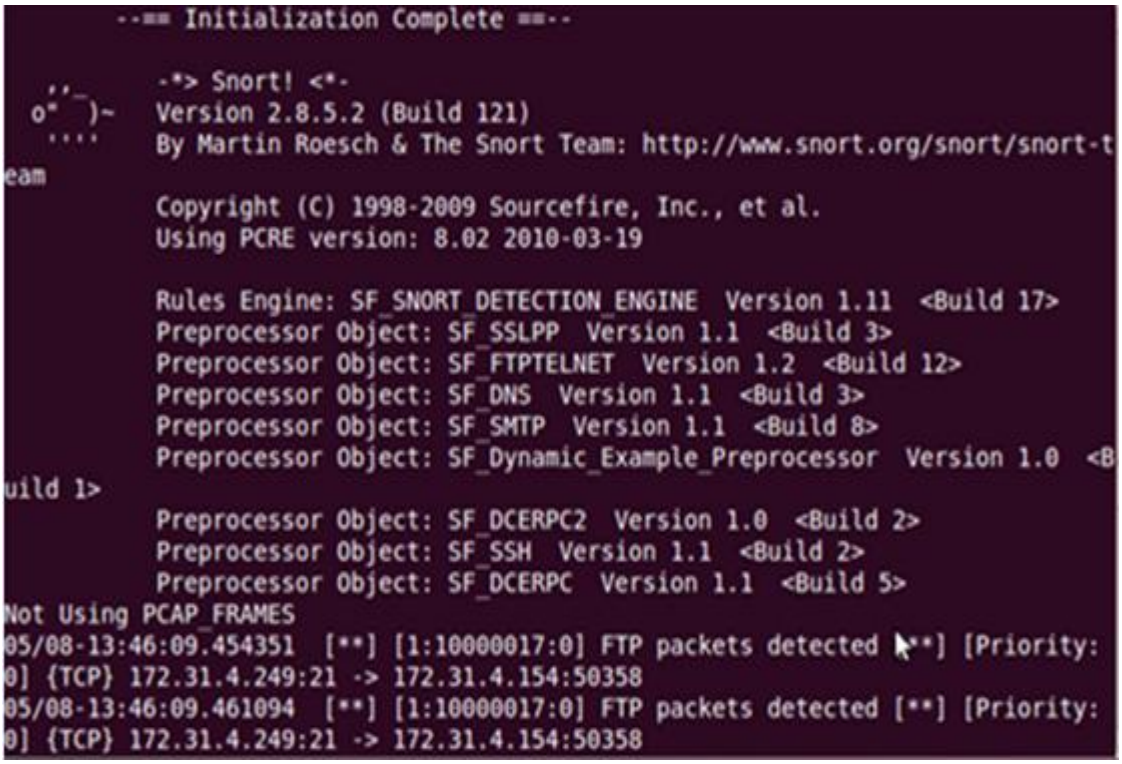
Snort is able to detect also on the basis of port number. Here in the example 21 is the port no for TCP protocol for FTP application. So the rule detects for any FTP packet on network on TCP

Alert tcp $HOME_NET 21 -> $EXTERNAL_NET any (msg:"ftp packets detected"; sid: 10000017 ;)

For capturing FTP packets vsftpd server is used where vsftpd, stands for "Very Secure FTP Daemon", is an FTP server for Unix-like systems, including Linux.

Run Snort using: *Snort –A console –c snort.conf –r ftp.pcap.*

Where –r used for reading from a file and ftp.pcap file stores the captured FTP packets. The alerts for FTP packets are shown in Snapshot 5.5.



Snapshot 5.5: Alerts for FTP packets

- **Understanding Standard Alert Output**

  The alerts in Snort are in the following form. The components in the alert are explained as:

  [*] [1:123456:0] someone opened google website [* *]*

  i. The first number is the Generator ID; this tells the user what component of Snort generated this alert. Gid 1 is associated with the rules subsystem and various gids over 100 are designated for specific preprocessors and the decoder. See etc/generators in the source tree for the current generator ids in use.

  ii. The second number is the Snort ID (sometimes referred to as Signature ID). For a list of preprocessor SIDs, please see etc/gen-msg.map. Rule-based SIDs are written directly into the rules with the *sid* option. In this case 123456 is represented as a sid given by user to create the rule for google.

iii. The third number is the revision ID. This number is primarily used when writing signatures, as each rendition of the rule should increment this number with the *rev* option. The rev keyword is used to uniquely identify revisions of Snort rules. Revisions, along with Snort rule id's, allow signatures and descriptions to be refined and replaced with updated information. This option should be used with the sid keyword.

5.3.2 Milestones covered on Window 7

Milestones covered in Window7 include running Snort in all its intrusion detection modes. Snort is run in real time in all the modes. Various experiments performed on Ubuntu include the following.

- **Run Snort in all operation modes**

Snort has basically four modes of operation but if Snort is operated in intrusion detection mode it has three modes. When Snort works in an IDS mode, it captures packets using Libpcap or Winpcap. Inline mode of the Snort does not come as an intrusion detection system. It works as an intrusion prevention system. It can also drop or reject the packets as per the requirement. It captures the packets using iptables rather than using libpcap or Winpcap. Following are the four operation modes supported by Snort.

i. Sniffer mode
ii. Packet logger mode
iii. NIDS mode
iv. Inline mode

All the modes are performed on Window7 on real time traffic and also on captured data and are shown with the help of the snapshots.

The modes are as shown with the experimental results as:

i. **Experimenting with Sniffer mode:** Simply reads the packet off the network and displays them in a continuous stream on the console. Options to run in a sniffer mode are:

- **Case 1:** If only the TCP/IP headers are to be printed on the console window case 1 can be used.

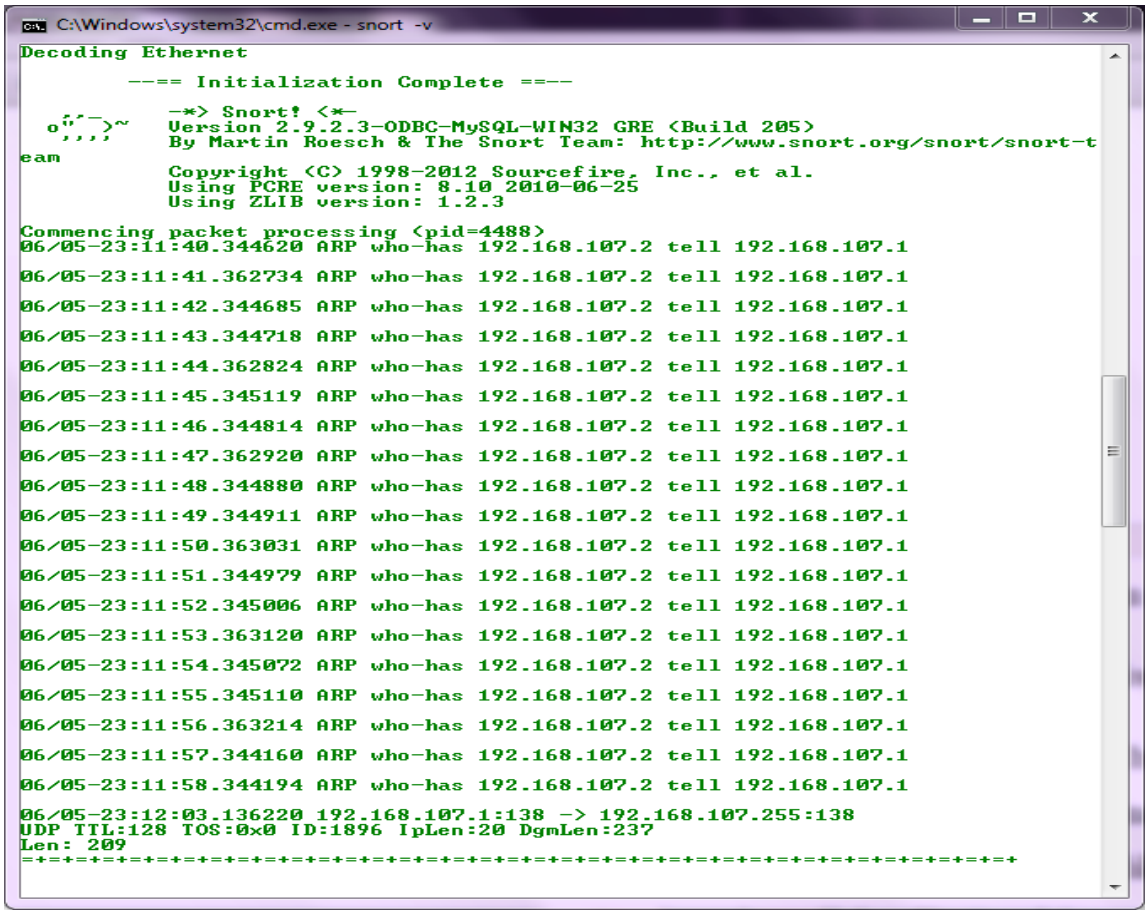
  a) Procedure: Open the command prompt in Windows and perform the following steps in command prompt:

  cd

  cd c:\etc\snort\bin

  snort –v

  b) Output: The output is shown in Snapshot 5.6:



Snapshot 5.6: Packets containing TCP/IP header

- **Case 2:** If the TCP/IP headers are to be printed along with the application data the user need to follow the second case. The packets will be displayed on the console.

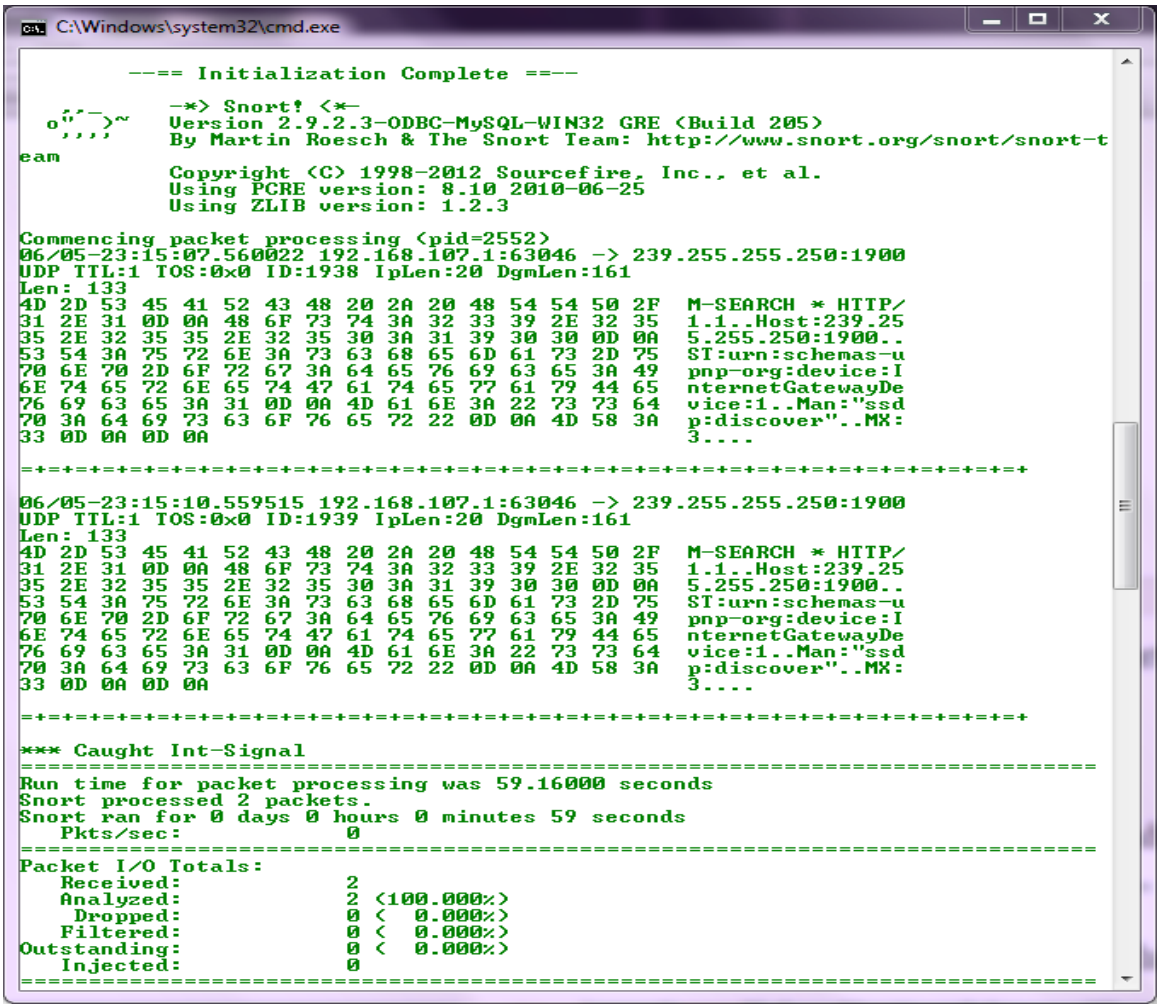
    a) Procedure: Open the command prompt in Windows and perform the following steps in command prompt:

    cd

    cd c:\etc\snort\bin

    snort –vd

    b) Output: The output is shown in Snapshot 5.7.



Snapshot 5.7: Packets containing TCP/IP headers & application data

- **Case 3:** If data link layer contents are to be printed along with the TCP/IP headers and application data the user need to follow the third case. The packets will be displayed on the console.

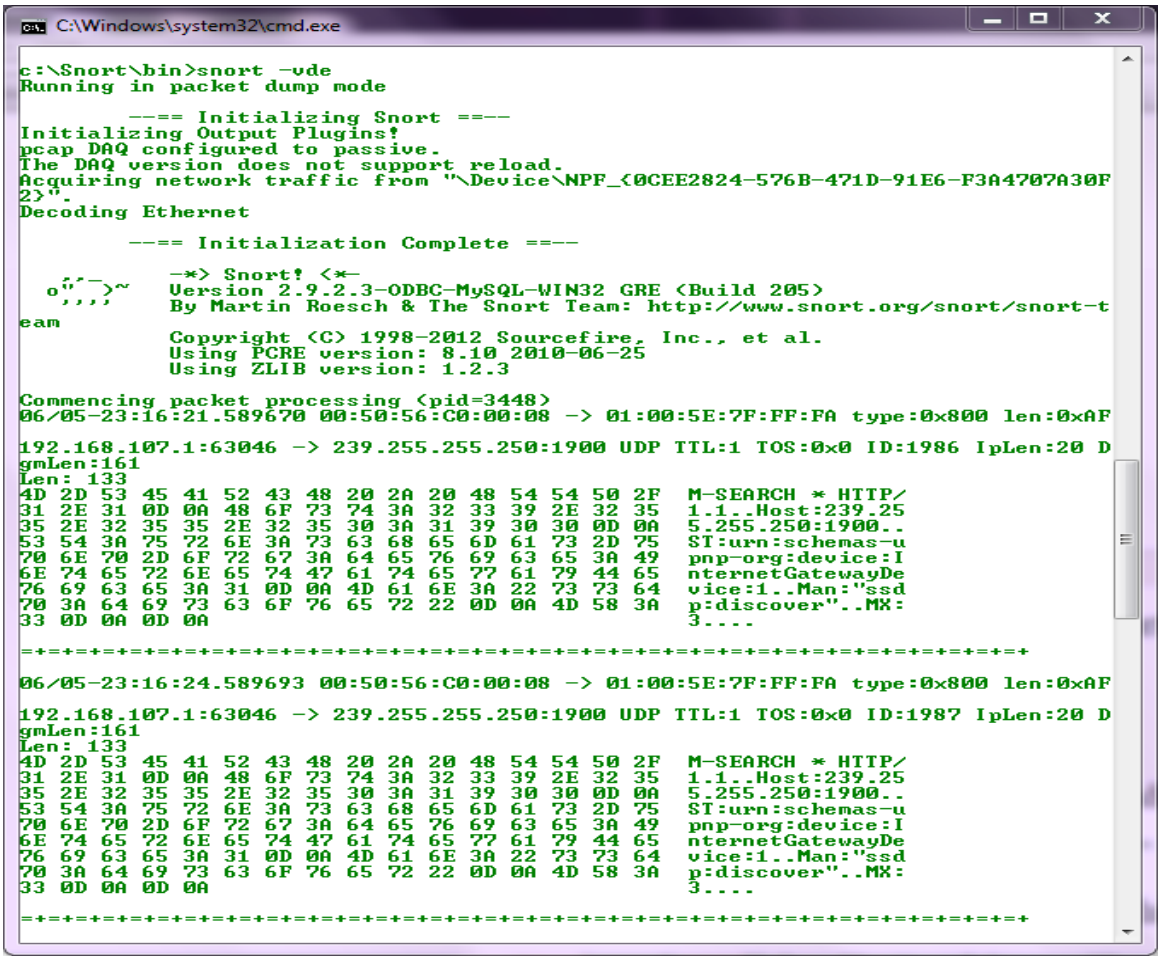
  a) Procedure: Open the command prompt in Windows and perform the following steps in command prompt

  cd

  cd c:\etc\snort\bin

  snort –vde

  b) Output: The output is shown in Snapshot 5.8.



. **Snapshot 5.8: Packets containing TCP/IP headers, application data & data link layer contents.**

ii. **Experimenting with Packet logger mode:** Logs the packets to the specified directory in disk. Options to run snort in packet logger mode are:

- **Case:** If one wants to log the packets in some directory for example in this case the directory c:\snort\log one can use this case.

    a) Procedure: Open the command prompt in Windows and perform the following steps in command prompt:

    cd

    cd c:\snort\bin

    snort –dev –l c:\snort\log

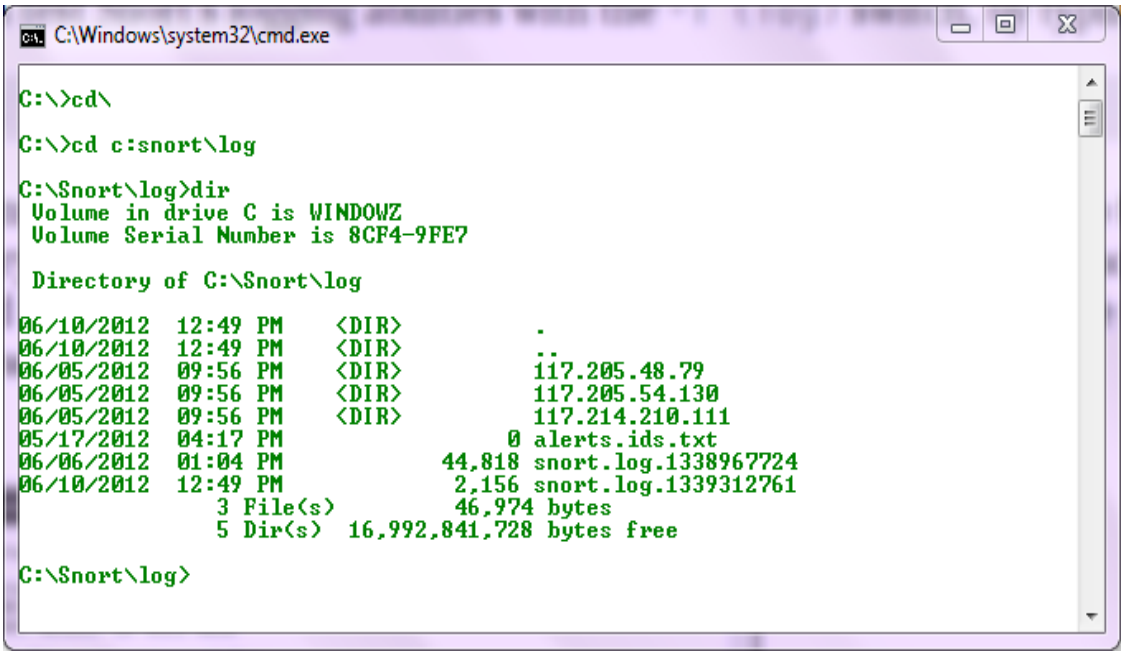
    To check the logged packets follow the procedure:

    b) Procedure to check output: Open the command prompt in Windows and perform the following steps in command prompt:

    cd

    cd c:snort\log\dir

    c) Output: The output is shown in Snapshot 5.9.



Snapshot 5.9: Check Snort\log Directory

iii. **Experimenting with Network Intrusion Detection Mode (NIDS):** Most complex and configurable mode. It allows Snort to analyze network traffic against a user defined rule set and performs actions based on detections.

- **CASE:** If one wants to display the alerts generated on console window if any network activity matches against the rules defined in Snort.

  a) Procedure: Open the command prompt in Windows and perform the following steps in command prompt:

  cd

  cd c:\etc\snort\bin

  snort –A console – i2 -c c:\snort\etc\snort.conf –l c:\snort\log – K ascii

  The procedure will display alerts on console and log packets in c:\Snort\log.

  b) Output: The output is shown in Snapshot 5.10.

```
Commencing packet processing (pid=5660)
05/29-19:12:35.659410  [**] [1:100000013:0] ô**TCP packets entering our network*
*ö [**] [Priority: 0] {TCP} 117.205.52.85:50754 -> 10.1.1.155:1433
05/29-19:12:41.659300  [**] [1:100000013:0] ô**TCP packets entering our network*
*ö [**] [Priority: 0] {TCP} 117.205.52.85:50754 -> 10.1.1.155:1433
05/29-19:12:53.661688  [**] [1:100000013:0] ô**TCP packets entering our network*
*ö [**] [Priority: 0] {TCP} 117.205.52.85:50755 -> 50.18.46.39:587
05/29-19:12:53.836769  [**] [1:100000013:0] ô**TCP packets entering our network*
*ö [**] [Priority: 0] {TCP} 117.205.52.85:50756 -> 74.125.77.109:25
*** Caught Int-Signal
===============================================================================
Run time for packet processing was 20.21000 seconds
Snort processed 17 packets.
Snort ran for 0 days 0 hours 0 minutes 20 seconds
   Pkts/sec:           0
===============================================================================
Packet I/O Totals:
   Received:           17
   Analyzed:           17 (100.000%)
    Dropped:            0 (  0.000%)
   Filtered:            0 (  0.000%)
Outstanding:            0 (  0.000%)
   Injected:            0
===============================================================================
```

Snapshot 5.10: Alert for TCP packets

5.3.3 *G*-Snort

G-Snort is a graphical user interface for Snort rule creation and execution. It has been created using .NET Window form application on Window 7. It helps the user to use Snort in an easy way. It prevents user from using command line interface and to write typical commands to make Snort work, rather it is graphical user interface to work with Snort. In *G*-Snort different operational modes of Snort are provided. They can be run with an ease by just clicking on the respective buttons provided in the application. It also provides different supports like documentations which make it easy to work with application also. Following are the functionalities provided by *G*-Snort:

i. User can create any rule according to the requirement.

ii. Rule will be included in snort.conf file automatically after the rule is created.

iii. User can run Snort in all the possible modes just by clicking on few buttons. Following modes are provided in *G*-Snort.
   a) Sniffer mode.
   b) Packet logger mode.
   c) NIDS mode.

iv. User can read various documents in *G*-Snort to have better understanding about Snort and its rules.

v. User can even read about how to use *G*-Snort.

The functionality of *G*-Snort is shown in with the help of snapshots in the following pages.

i. ***G*-Snort Home:** Home page of snort graphical user interface is shown below in snapshot 5.11. It contains following tabs:

a) Make rule: This tab brings us to the page where the snort rules can be created.

b) Run Snort: To run the snort in different modes as described in section 3.5.

c) Read-me: The document page contains the following documents:

i. The document for providing the information about various keywords used in Snort rule making.

ii. Some sample rules are provided to help the users to make rules.

iii. The information about how to use the *G*-Snort front end application document is provided.

iv. The document for providing the information regarding the modes of snort to help the user run the Snort.

Snapshot 5.11: G-Snort Home

ii. **Make rule (Scenario 1):** Snapshot 5.12 is showing the Snort rule making window. Here all the options related to the Snort rules are provided.

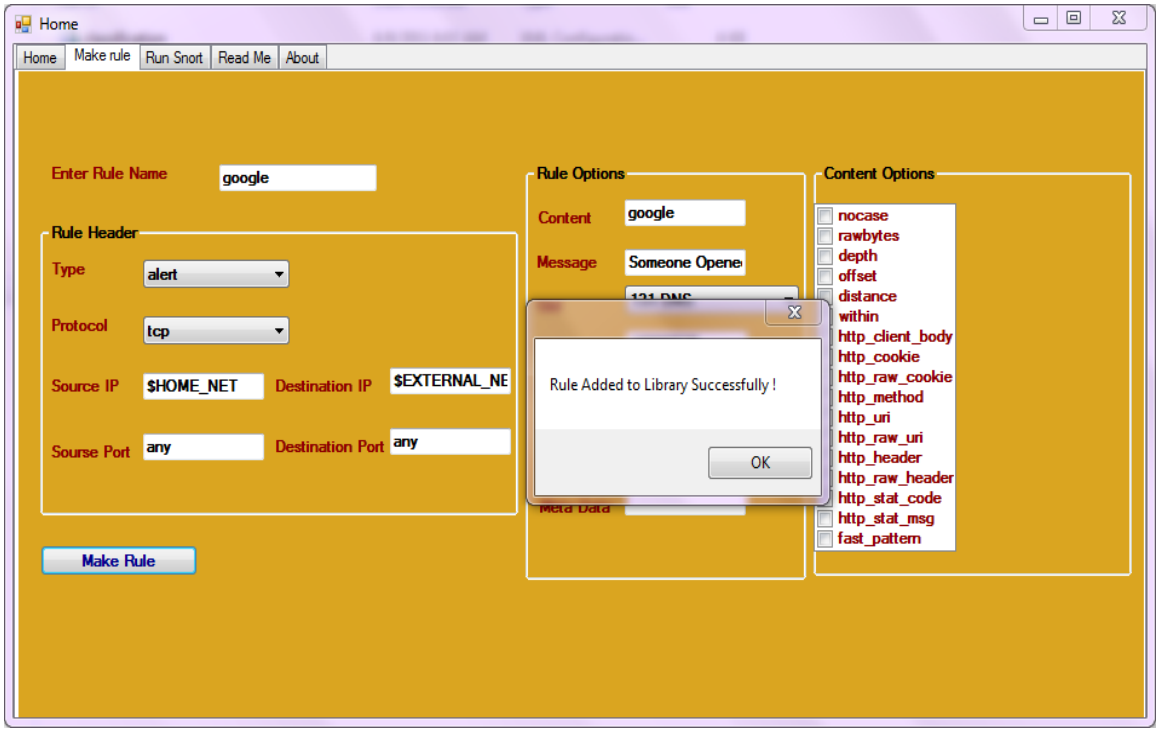
a) User can select the option and can fill the desired value as per their requirement.

b) Left hand side of the window shows the rule header part of the Snort rule all of which are the mandatory field. If any of the field is left unfilled the rule will not be created.

c) The window shows the following rule

```
alert tcp $HOME_NET any -> $EXTERNAL_NET any
(content:"google"; msg:"Some One Opens Google" ;
gid:116; sid:10000028;)
```

d) The message box in the snapshot is representing that the rule has been successfully created. This screen will by default add rule named "google.rules" in c:\snort\rules and will include the rule in c:\snort\etc\snort.conf.



Snapshot 5.12: G-Snort Make-Rule (1)

iii. **Make rule (Scenario 2):** Snapshot 5.13 shows the window which shows the error message when any of the header field is left unfilled.

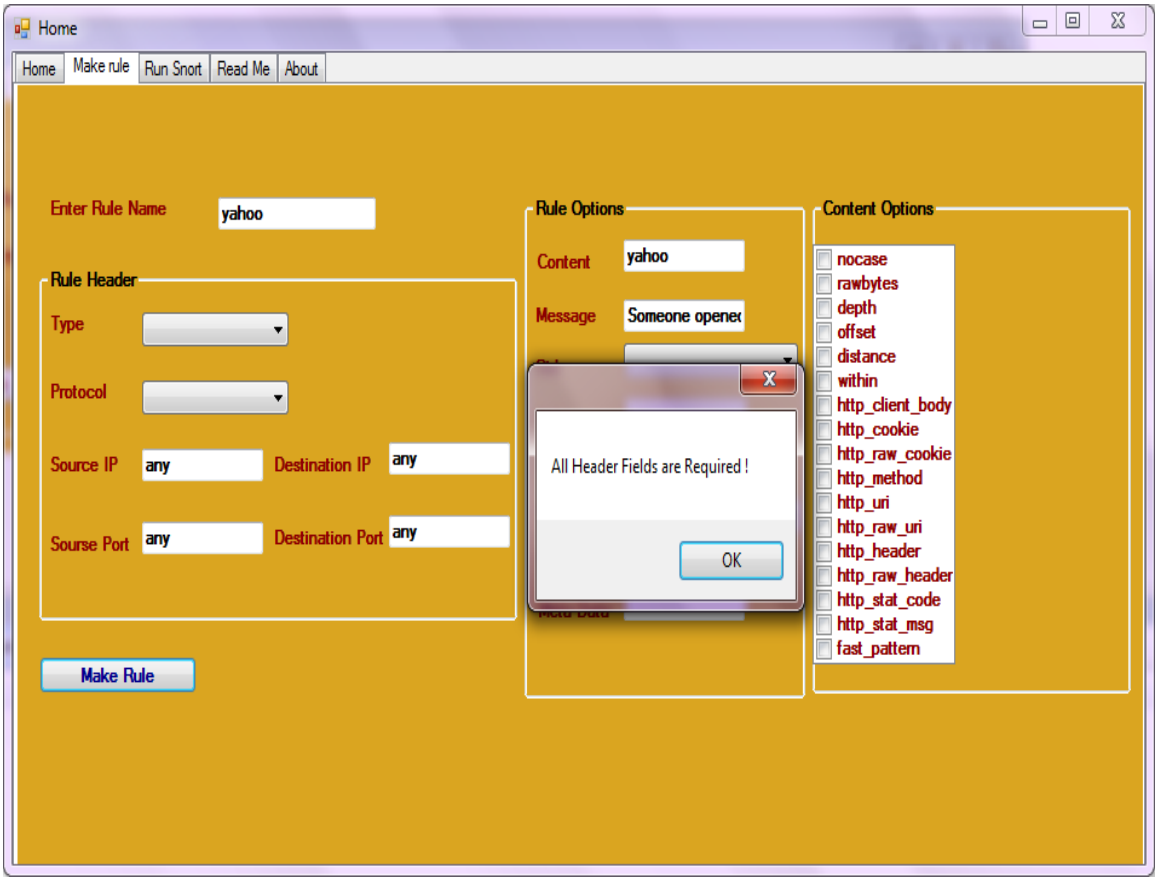
a) Snort rules are divided into two parts: Snort header, Snort options.

b) Left hand side of the window shows the header part of the Snort rule and right hand side is the option part.

c) All the components in the header part are mandatory. Snort is not able to work if any of the field is missing.

d) The snapshot shows that user has not filled "rule name", "type", "protocol", in the rule making window, so the error message "*All header fields are required*" is displayed be *G*-Snort.

e) However if only one field is missing still it will be an error.



Snapshot 5.13: G-Snort Make-Rule (2)

iv. **Make rule (Scenario 3): S**napshot 5.14 shows the error message if user is trying to create the rule which already exists.

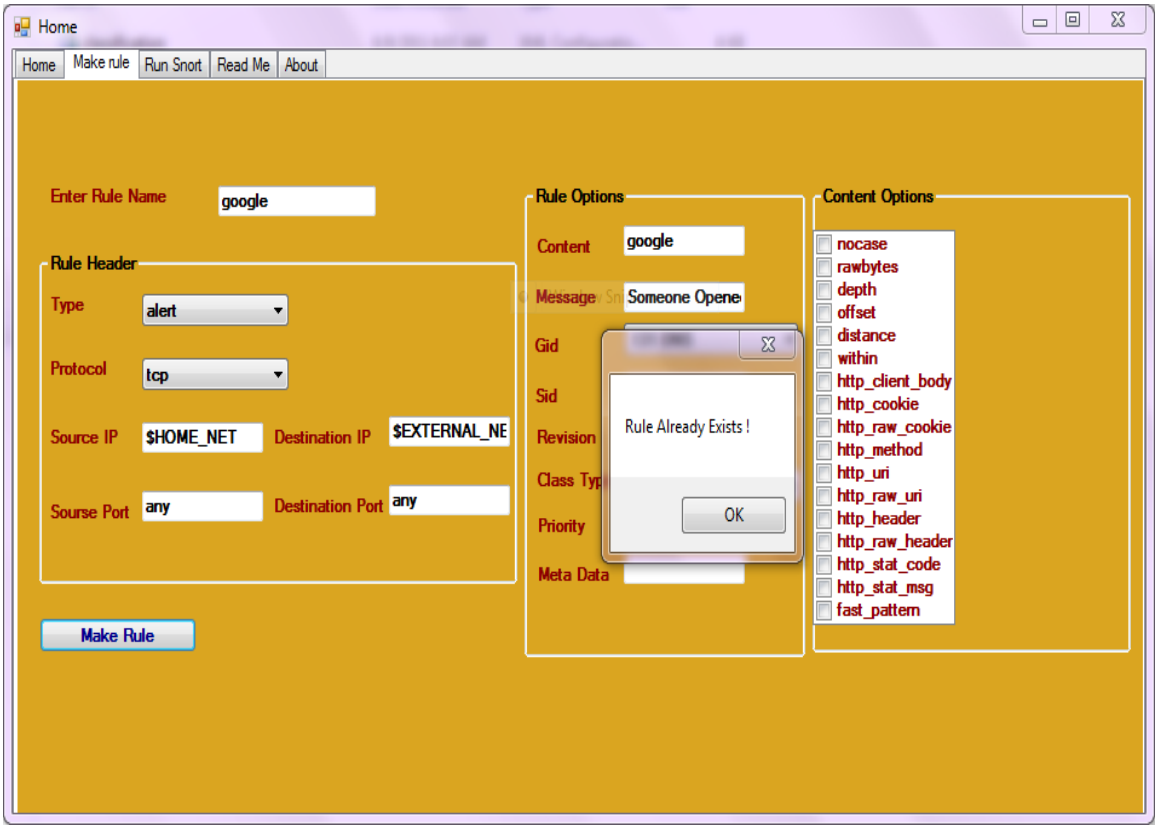
a) User has tried to create the following Snort rule:

```
alert tcp $HOME_NET any -> $EXTERNAL_NET any (content:"google"; msg:"Some One Opens Google" ; gid:116; sid:10000028;)
```

b) The user has created the same rule as created in the In the Make rule (Scenario 2).

c) But *G*-Snort does not allows the user to create the rule which is having the same name as that of some previous rule.

d) So in the following snapshot a message box is showing an error message "*rule already exists.*" when user is trying to create the same rule.



Snapshot 5.14: G-Snort Make-Rule (3)

v. **Run Snort (scenario 1):** Snapshot 5.15 shows various operational modes in Snort:

a) Snort NIDS has three operation modes named “Sniffer mode”, “Packet logger mode,”, “NIDS mode”.

b) Sniffer mode contains the dropdown list which contains options in Sniffer mode

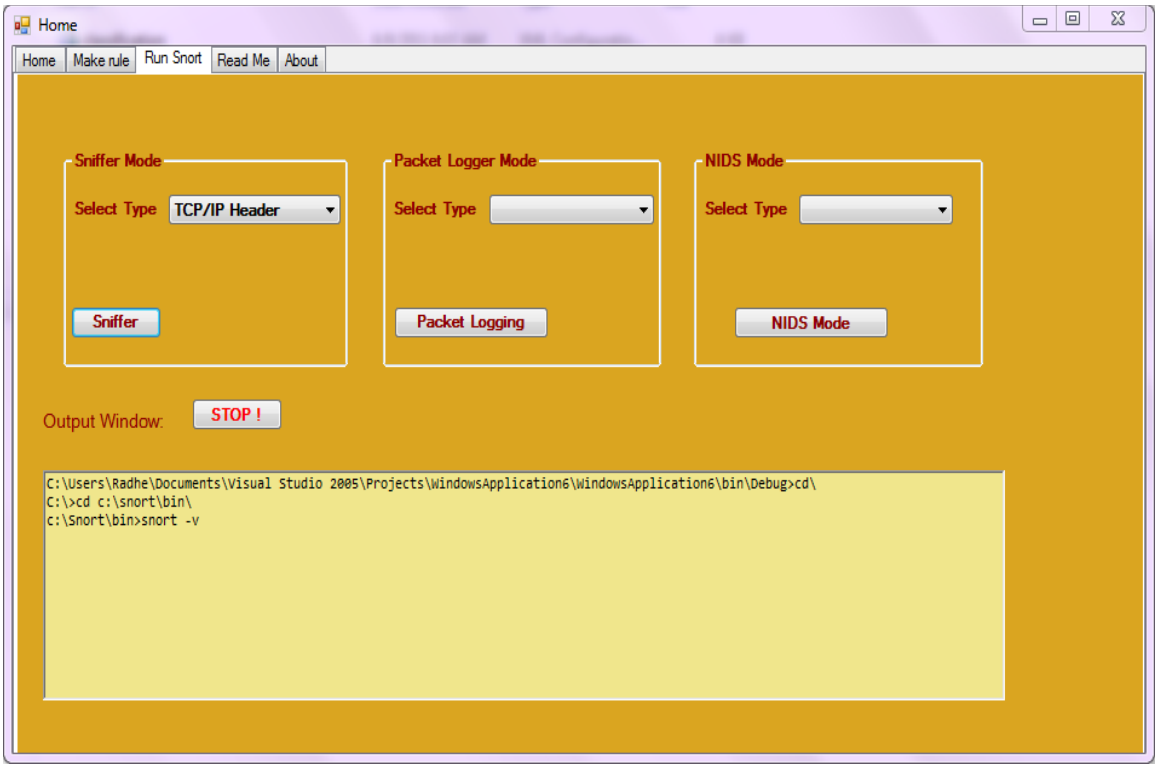
c) Dropdown list is provided for all the three modes which contains further options in the modes.

d) The options can be selected using dropdown list.

e) After selecting an option and clicking on the respective buttons the results are displayed on the output window given at top of the form.

f) The stop button is provided so as to stop the process.

g) If the user wants to move at any page from this page then the tabs on the top off the page helps the user to move at any page.



Snapshot 5.15: G-Snort Run Snort (1)

vi. **Run Snort (scenario 2):** Snapshot 5.16 is displaying the output of Snort :

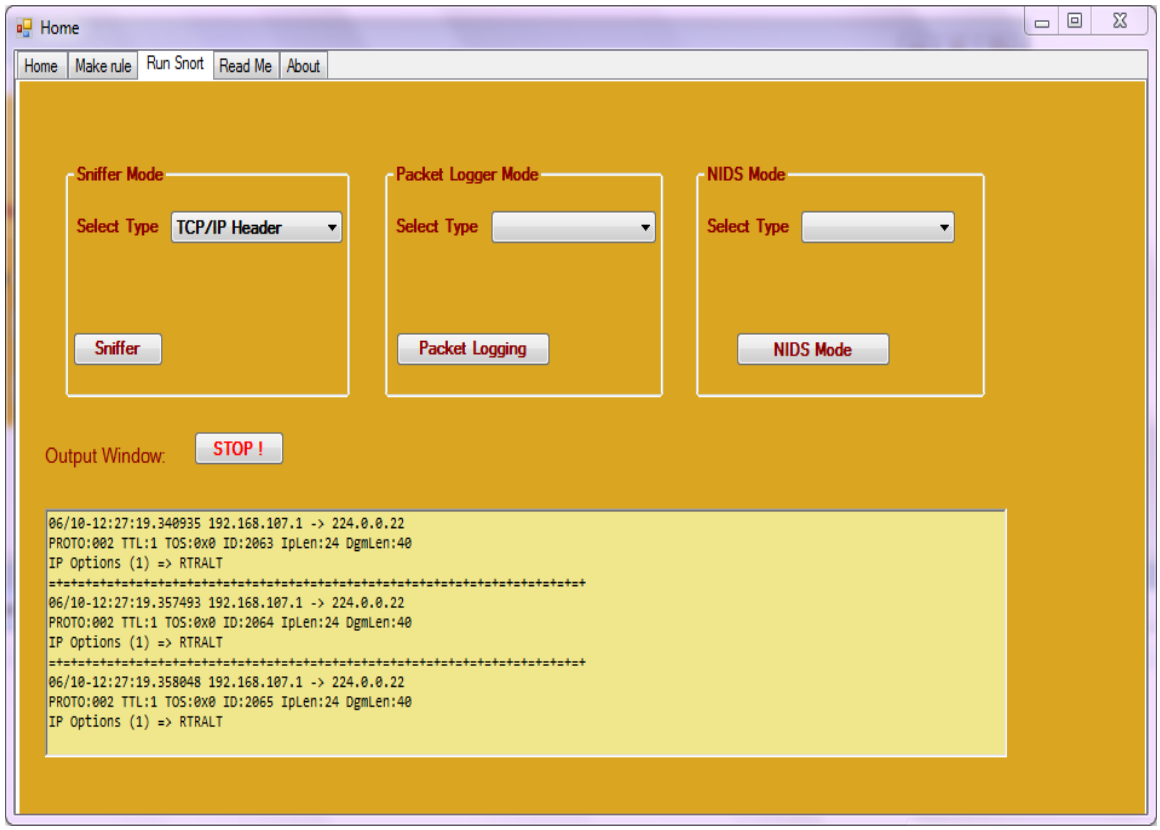
a) The options in the sniffer mode "display TCP/ID header on the screen" is selected from the dropdown list.

b) The sniffer mode button provided in the form is clicked.

c) The results of sniffer mode are shown in the rich textbox provided at the bottom of the page.

d) TCP/IP headers of the packets coming on the network are shown in the output window.

e) The packets can be stopped by clicking on the button "STOP" provided in the middle of the form.



Snapshot 5.16: G-Snort Run Snort (2)

vii. **Read-me: S**napshot 5.17 shows sample rules provided by *G*-Snort to help the user to have understanding of rule:

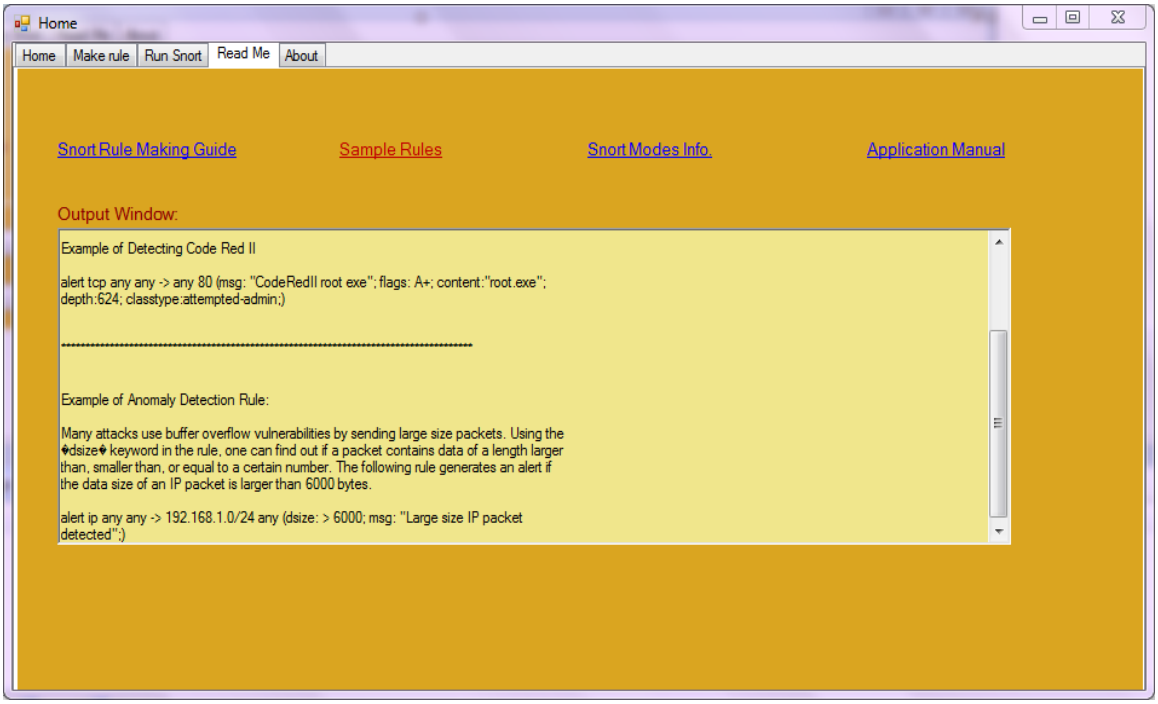
a) *G*-Snort has an additional advantage that it provides the users an additional information regarding snort. Following documents are provided in *G*-Snort :

b) Snort rule making guide which provides the rules and guidelines about the rule making process.

c) Sample rules which provides the users to help the user to have better understanding of the rule.

d) Snort modes information provides the information about the operational modes of the Snort.

e) Application manual provides the guidelines for how to use the G-Snort application.



Snapshot 5.17: G-Snort Read Me

In this chapter Snort implementation has been discussed and also its rule making process. Finally *G*-Snort is discussed which is a graphical user interface for Snort rule making and execution.

Chapter 6

Conclusion and Future Scope

6.1 Conclusion

The objective of this thesis is to analyze a network based intrusion detection called snort and to make rules to detect any malicious activity.

To accomplish the objective, first the stages of an attack are analyzed to understand at which stages an attack can be detected. Then firewall, network-based antivirus system and honeypot technologies are examined to determine when, how and why they are used. Also, the limitations of those technologies is defined which are the cause to the new IDS technology. Then a detailed study on intrusion detection systems was carried out so that the main task to explore the Snort which is an intrusion detection system can be better understood.

Snort system's detailed study is performed and then installed it on Ubuntu and Window7. Explored Snort by running it in different modes which includes sniffer mode, packet logger mode, and network intrusion detection system mode. Where as one more mode of Snort is there called inline mode but in this mode Snort does not work as an Intrusion Detection System. So the work is done on the above three modes. Then rules were created based on different options which includes 'protocol', 'content' and 'port number'. Finally a graphical user interface called *G*-Snort was created to make the rules and run Snort in its different operation modes. The graphical user interface made it easy to make rules in Snort. It also let us run Snort without remembering the typical commands. It was designed by considering all the user requirements. It provides all kinds of documentation support for the better understanding of user. *G*-Snort has the following advantages:

- Different documents are provided about Snort rules and Snort modes to help users.
- Documents about using the front end application *G*-Snort is also provided for help.
- An easy solution to Snort making and execution.

6.2 Future Scope

In this thesis we showed the Snort NIDS and its rule making process. The rule is applied on the interface traffic as well as the captured packets. As we have seen the rule making and executing process requires a lot of manual tasks and time. So to avoid it a front end for rule making is shown which minimizes lots of our tasks and very helpful also in the situation where one has a limited amount of knowledge about snort rules.

Our future direction will therefore be to develop a more vast front end which may even shows outputs in different formats, and may also applies statistics on the network data. One of the future directions is to find out the signatures for new worms and virus and to make signatures to detect those through Snort.

Snort can also act as an IPS when operated in inline mode. One of the future works can be to make Snort run in IPS mode and prevent the intrusions in the system.

References

[1] H. Redwan, Ki-Hyung Kim, "Survey of Security Requirements, Attacks and Network Integration in Wireless Mesh Networks," in Proc. Japan-China Joint Workshop on Frontier of Computer Science and Technology, Dec. 2008, pp. 3-9.

[2] W. Stallings, Cryptography and Network security: Security Services, 4th Ed., Delhi: Prentice Hall, 2007.

[3] C.C. Zou, N. Duffield, D. Towsley, and W. Gong, "Adaptive Defense Against Various Network Attacks," IEEE Journal on selected areas in communications, vol. 24, No. 10, Oct. 2006.

[4] S. Ansari, S.G. Rajeev and H.S. Chandrashekar, "Packet Sniffing: A Brief Introduction." IEEE potentials, vol. 21, pp. 17-19, Jan. 2003.

[5] Ming-Wei Wu, Yi-Min Wang, Sy-Yen Kuo, and Yennun Huang, "Self-Healing Spyware: Detection, and Remediation." IEEE Transactions on Reliability, vol. 56, pp. 588-596, Dec. 2007.

[6] K. Johansson, "Offensive Operations Model", KSAJ Inc, Version: 1 .0 Public Release, Posted: http://www.penetrationtest.com, August **7,** 2001.

[7] S. Nassar, A.E. Sayed, N. Aiad, "Improve the Network Performance By using Parallel Firewalls," in Proc. of 6th International Conference on Networked Computing, May 2010, pp. 1-5.

[8] X. Jhang, C. Li, W. Zheng, "Intrusion Prevention System Design." in Proc. of 4th International Conference on Computer and Information Technology, pp. 386-390, Sept. 2004.

[9] A. S. Ashoor and S. Gore, "Importance of Intrusion Detection system (IDS)". International Journal of Scientific and Engineering Research, vol. 2, no. 1, pp.1-4, Jan-2011.

[10] W. Lee, Salvatore J. Stolfo, and Kui W. Mok, "A Data Mining Framework for Adaptive Intrusion Detection," in Proc. of the lEEE Symposium on Security and Privacy,1999, pp.I20-132.

[11] Harek Haugerud, "Intrusion detection and firewall security," Available: http://www.iu.hio.no/teaching/materials/MS004A/html/pictures/ids.png.

[12] V. Visoottiviseth, U. Jaralrungroj, E. Phoomrungraungsuk, P. Kultanon, "Distributed Honeypot log management and visualization of attacker geographical distribution," in Proc. of Eighth International Joint Conference on Computer Science and Software Engineering (JCSSE), May 2011, pp. 23-28.

[13] L. Spitzner, "The Honeynet Project: Trapping the Hackers," IEEE Security & Privacy, vol. 1, pp. 15-23, Apr. 2003.

[14] Fortinet, Inc. "Improving Network Protection and Performance with Network-Based Antivirus Technology," White paper, Oct. 2002.

[15] A. Lazarevic, V. Kumar and J. Srivastava, "Managing Cyber Threats: Issues, Approaches and Challenges, chapter: A survey of Intrusion Detection techniques," Kluwer Academic Publishers, 2005

[16] P.Cisar and S. M. Cisar, "Intrusion Detection–One of the Security Methods." in Proc. of 6th International Symp. on Intelligent Systems and Informatics, 2008, pp. 1-6.

[17] A. Hay, D. Cid, R. Bray, "OSSEC Host-Based Intrusion Detection System," Syngress Publishing, Inc., 2008.

[18] W. W. Stames, "Integrity Assessment Tools: Fundamental Protection for Business Critical Systems, Data and Applications," in Proc. of 22nd International Conference on Information Technology Interfaces, 2000, pp. 465-470.

[19] M. Ahmed, R. Pal, M.M. Hossain, M.A.N. Bikas, M.K. Hassan, "NIDS: A Network Based Approach to Intrusion Detection and Prevention," in Proc. of 9th International Association of Computer Science and Information Technology, Apr. 2009, pp. 141-144.

[20] V. Paxson, "Bro: A System for Detecting Network Intruders in Real-Time," in Proceedings of 7th USENIX Security Symposium, San Antonio, TX, January 1998, pp. 2435-2463, 14 Dec. 1999.

[21] D. J. Brown, B. Suckow, and T. Wang, "A Survey of Intrusion Detection Systems," 2002.

[22] White paper, "Intrusion Detection: A Survey," ch.2, DAAD19-01, NSF, 2002.

[23] F.Sabahi, A.Movaghar, "Intrusion Detection: A Survey," in Proc. of 3rd International Conference on Systems and Networks Communications, 2008, pp. 23-26.

[24] T. Wang and X.D. Yang, "Intrusion Detector: A Software Platform for Testing Network Intrusion Detection Algorithms," 2001.

[25] M.E. Kuhl, M. Sudit, J. Kistner, and K. Costantini, "Cyber attack modeling and simulation for network security analysis," in Proc. IEEE Simulation Conf., Dec. 2007, pp. 1180- 1188.

[26] S. Sorensen, "Intrusion Detection and Prevention Protecting Your Network From Attacks," White Paper, 2006.

[27] M. Roesch, "Snort- Lightweight Intrusion Detection for Networks," In Proc. of 13th USENIX conference on System administration LISA '99, CA, USA, pp. 229-238, Nov. 1999.

[28] D. Stiawan, A.H. Abdullah, M.Y. Idris, "The trends of Intrusion Prevention System network," in Proc. 2nd International Conference on Education Technology and Computer (ICETC), Jun. 2010, pp. V4-217-V4-221.

[29] S. Niccolini, R. G. Garroppo, S. Giordano, G. Risi, S. Ventura, "SIP Intrusion Detection and Prevention: Recommendations and Prototype Implementation." in Proc. of 1st IEEE Workshop on VoIP Management and Security, 2006, pp. 47-52.

[30] M. A. Qadeer, M. Zahid, A. Iqbal, M.R. Siddiqui, "Network Traffic Analysis and Intrusion Detection using Packet Sniffer." in Proc. ICCSN, 2010, pp. 313-317.

[31] W. Xuren, H. Famei, "An Implement of Broadband Network Monitoring System Based on Libnids and Winpcap," in Proc. International Conference on New Trends in Information and Service Science, 2009, pp. 812-814.

[32] L. Xuanmin, S. Chang, Moses, G. Jingyuan, "Research on IP Address Replacement Technology Based on Iptables," 7th International Conference on Wireless Communications, Networking and Mobile Computing , Sept. 2011, pp. 1-3.

[33] K. Salah, A. Qahtan, "Boosting Throughput of Snort NIDS Under Linux," in Proc. of International Conference on Innovations in Information Technology, Dec. 2008, pp. 643-647.

[34] Snort Team, "Snort Users manual," Available at: http://www.snort.org/assets/166/snort_manual.pdf , Dec. 7, 2011.

[35] S. Sen, "Performance Characterization & Improvement of Snort as an IDS."

[36] A. Machie, J. Roculan, R. Russell, and M. V. Velzen, "Nimda Worm Analysis," Tech. Rep., Incident Analysis Report, Security Focus, Sept. 2001.

[37] K. Salah A. Kahtani, "Improving Snort performance under Linux," IET Communications, vol. 3, pp.1883-1895, Apr. 2009.

[38] Y. Chen, K. Hwang, W.S. Ku, "Collaborative Detection of DDoS Attacks over Multiple Network Domains," IEEE Transactions on Parallel and Distributed Systems, vol. 18, pp-1649-1662, Dec. 2007.

[39] V. Jacobson, C. Leres, and S. McCanne, TCPDUMP, Version 2.2., Lawrence Berkeley Laboratory, University of California, Berkeley, CA.

[40] Q. Guangzhi et al. "Abnormality metrics to detect and protect against network attacks." in Proc. of the IEEE/ACS International Conf. on Pervasive Services, 2004, pp. 105-111.

List of Publications

1. Isha Singla, Sanmeet Kaur, "Research Paper Showing Snort rule Execution and front End for Snort Rule Making", International Conference on Advanced Computing Technologies, June 2012 at Gurukul Vidyapeeth, Chandigarh.
2. Isha Singla, Sanmeet Kaur, "Intrusion Detection System Classification: A Survey", International Conference on Advanced Computing Technologies, June 2012 at Gurukul Vidyapeeth, Chandigarh.