A Hybrid approach using Signature and Anomaly Detection to detect network Intrusions

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

Master of Technology
in
Computer Science and Applications

Submitted By

Tejvir Kaur

(601103026)

Under the supervision of

Ms. Sanmeet Kaur

Assistant Professor

SMCA

School of Mathematics and Computer Applications

THAPAR UNIVERSITY

PATIALA – 147004

June 2013

CERTIFICATE

I hereby certify that the work which is being presented in the thesis entitled, **"A Hybrid approach using Signature and Anomaly Detection to detect network Intrusions"**, in partial fulfillment of the requirements for the award of degree of Master of Technology in Computer Science and Applications submitted in School of Mathematics and Computer Applications of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of Ms. 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.

Tejvir Kaur.
(Tejvir Kaur)

601103026

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

Sanmeet Kaur

(Ms. Sanmeet Kaur)
Assistant Professor
SMCA

Countersigned by

(Dr. Rajesh Kumar)
Head,
SMCA Department,
Thapar University,
Patiala.

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

ACKNOWLEDGEMENT

I would like to express my deepest appreciation to Ms. Sanmeet Kaur, my mentor and thesis supervisor for providing me with many valuable ideas and insights. She has constantly supported me through the thesis work. The successful completion of my thesis work is all due to her guidance and motivation. It was an immense pleasure for me to work under her supervision.

I am also thankful to Dr. S.S. Bhatia, Head, School of Mathematics and Computer Applications 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 love, support and enthusiastic encouragement without which I could not complete this thesis. I also want to thank my friends for optimism and moral support throughout the completion of this work. Finally I thank the Almighty who gave me the strength to complete this work.

Tejvir Kaur

601103026

ABSTRACT

Network Security has become a crucial issue for most organizations in the recent past. Mostly discussions on security include the tools and methods that can be deployed to protect and defend the networks. The use of network security tools have increased over the years due to increase in security threats. Many methods have been developed to secure computer networks and communication over the Internet. Intrusion detection method is one such method which has gained importance over the past few years. An Intrusion Detection System gathers and analyzes information from various areas within a computer or a network to identify possible security breaches.

There are different techniques to detect intrusions and these techniques are discussed in the thesis. On the broader level, there are two techniques that are for detecting Intrusions viz. signature detection and anomaly detection. Signature detection detect intrusions by matching the network traffic with database of stored signatures and anomaly detection looks for behaviour deviating from normal or common behaviour for detecting intrusions. The signature detection can detect well known attacks giving low false positives whereas anomaly detection can detect new attacks but also has high false positive rate.

The primary objective of the thesis work is to combine both these techniques. A combined approach of anomaly detection and signature detection is compared with signature detection. The DARPA IDS Evaluation dataset is used for this purpose. The pros and cons of using the DARPA IDS Evaluation dataset are also discussed. Experimental evaluation shows that the combined approach of anomaly and signature detection gives better performance. Furthermore, the attack related information stored by the systems during the experimentation is used to classify the network packets in the DARPA IDS Evaluation dataset. This is done to achieve the next goal of the thesis which comprises of analyzing this data on a machine learning tool. Finally the data is processed on a classification algorithm to obtain the results. The results show high percentage of correct classification.

Table of Contents

List of Figures

List of Tables

List of Snapshots

Chapter 1

Introduction

1.1 Introduction to Security

With the advent of Internet, personal computers and computer networks are becoming increasingly vulnerable to various kinds of attacks. Information has become like an asset that needs to be protected from attacks. Hackers make use of the security breaches present in the system or network to attack it. Due to security breaches, individual users and organisations get affected. Internet has completely changed the way the things were done in the past but the sensitive information sent over it has become vulnerable to various types of threats. Due to attack privacy can be violated and important data can be lost. The attacks are usually caused by a failure to implement security policies and failure of using of security tools that are readily available. Detection of attacks in the network traffic is one of the major goals of security. The following are the three major goals of security.

- **Confidentiality:** Information needs to be hidden from unauthorized access.
- **Integrity:** Information needs to be protected from unauthorized change.
- **Availability:** Information should be available to an authorized entity when it is needed [1].

1.2 Security Attacks

The three goals of security that are confidentiality, integrity and availability can be threatened by security attacks. The following section explains the security attacks [1] that threaten the goals of security.

1.2.1 Attacks Threatening Confidentiality

- **Snooping:** It means interception of data by unauthorized entity. This may be done to intercept the confidential data.
- **Traffic analysis:** By looking at the type of traffic, the type of information exchanged between the sender and the receiver could be extracted.

1.2.2 Attacks Threatening Integrity

- **Modification:** The attacker can change the information after getting access to the information for his/her own benefit.

- **Masquerading:** The attacker tries to impersonate somebody else to gather information from the user.
- **Replaying:** The attacker gets a copy of the message and later tries to replay it. The attacker replays the message for his/her own benefit.
- **Repudiation:** The sender of the message may later deny that he/she has sent the message and the receiver of the message may later deny that he/she has received the message.

1.2.3 Attacks Threatening Availability

- **Denial of service:** Denial of service attack halts the services of the system. The attacker may overload the system with bogus requests causing the system to crash.

1.3 Security Services

X.800 defines a security as a service. The security services [2] related to security goals and attacks are defined below.

1.3.1 Data Confidentiality

Confidentiality is the protection of transmitted data from passive attacks. In the context of a data transmission, several levels of protection can be identified. On the broader level, protection of all user data transmitted between two users over a period of time should be there. The narrower level of this service can be the protection of a single message or even specific fields within a message. Another aspect of confidentiality is the protection of traffic from analysis. This requires that an attacker should not be able to observe the source and destination, frequency, length or other characteristics of the traffic on a communications facility.

1.3.2 Data Integrity

A connection oriented integrity service is one that deals with a stream of messages. It assures that messages are received in the same form as sent with no duplication, insertion, modification, reordering or replays. The destruction of data is also covered under this service. Thus, the connection-oriented integrity service addresses both message modification and denial of service. On the other hand a connectionless integrity service is one that deals with individual messages. It generally provides protection against message modification only. The integrity services are related to active attacks. If a violation of integrity is detected, then the service simply reports this violation and other portion of software or human intervention is required to recover from the violation.

1.3.3 Authentication

The authentication service is concerned with assuring that a communication between two entities is authentic. In case of a single message, the function of the authentication service is to assure the recipient that the message is from the source that it claims to be from. The service should also make sure that the connection is not interfered with in such a way that a third party can masquerade as one of the two legitimate parties for the purposes of unauthorized transmission or reception.

Two specific authentication services are defined in X.800.

- **Peer entity authentication:** It provides the evidence for the identity of a peer entity in an association. This service attempts to provide confidence that an entity is not performing either a masquerade or an unauthorized replay of a previous connection.
- **Data origin authentication:** It provides the evidence for the source of a data unit. It does not provide protection against the duplication or modification of data units. This type of service supports applications like electronic mail where there are no prior interactions between the communicating entities.

1.3.4 Access Control

Access control means to limit and control the access to host systems and applications. To achieve this, each entity trying to gain access must first be authenticated so that access rights can be given to the individual.

1.3.5 Non Repudiation

Non repudiation prevents either sender or receiver from denying a transmitted message. Thus, when a message is sent, the receiver can prove that the alleged sender has sent the message. Similarly, when a message is received, the sender can prove that the alleged receiver has received the message [2].

1.4 Security Mechanisms

The following are the security mechanisms [1] recommended by X.800 to provide security services.

- **Encipherment:** Encipherment means hiding or covering data. The two techniques for used for encipherment are cryptography and steganography. This mechanism helps in providing data confidentiality.
- **Data integrity:** To check whether the data integrity is preserved or not, a checkvalue can be used. The checkvalue is sent by the sender along with the data. The receiver also

creates its own checkvalue and then receiver can compare its created checkvalue with the one received by the receiver to check the integrity of the message.

- **Digital Signature:** Data appended to a data unit that allows a recipient of the data unit to prove the source and integrity of the data unit and protect against forgery [2].
- **Authentication Exchange:** The messages are exchanged between entities to prove their identity to each other.
- **Traffic Padding:** Traffic padding is used to prevent the attacker from doing traffic analysis by inserting data that is not useful.
- **Routing Control:** The routes between the sender and the receiver are continuously changed so as to prevent the attacker from checking a particular route.
- **Notarization:** The communication between the sender and the receiver can be controlled by a third trusted party. This mechanism is used to prevent repudiation.
- **Access Control:** The passwords and PINs can be used as access control methods to allow user to access data.

1.5 Introduction to Intrusion Detection

Intrusion is the act of violating the security policy that pertains to an information system. Intrusion detection can be defined as the act of detecting actions that attempt to compromise the confidentiality, integrity or availability of a resource. Intrusion detection is the process of monitoring the events occurring in a computer system or network and analyzing them for signs of possible incidents, which are violations of computer security policies, acceptable use policies or standard security practices. Incidents have many causes such as malware (e.g., worms, spyware), attackers gaining unauthorized access to systems from the Internet, and authorized users of systems who misuse their privileges or attempt to gain additional privileges for which they are not authorized [3]. Intrusion detection provides the following functions.

- Monitoring and analysis of user and system activity.
- Auditing of system configurations and vulnerabilities.
- Assessing the integrity of critical system and data files.
- Statistical analysis of activity patterns based on the matching to known attacks.
- Abnormal activity analysis.
- Operating system audit [4].

1.6 Currently available network security solutions

The number of people connecting to the Internet is increasing very rapidly. The ease of use and the connectivity the Internet provides is highly useful but the risks involved and malicious intrusions are also increasing day by day. Exploitation of computer networks is getting more common. It is completely critical for business organization as well as individuals to protect their data from serious threats that would aim to steal their information. There are many security solutions available in the market. Some of them are like Firewall, Intrusion Detection System (IDS), Honeypot, antivirus which are explained below.

1.6.1 Firewall

A firewall is a combination of hardware and software that isolates an organization's internal network from other networks, allowing some packets to pass and blocking others. It functions to avoid unauthorized or illegal sessions established to the devices in the network areas it protects. Firewalls are configured to protect against unauthenticated interactive logins from the outside world. The firewall can be thought of as a pair of mechanisms: one which exists to block traffic, and the other which exists to permit traffic. Basically, numbers of firewalls can be deployed in the proper positions of the managed network for cooperative, integrated, and in-depth network security protection. Administrators that manage the firewalls have a have to be careful while setting the firewall rules [5]. The types of firewall are discussed below.

- **Packet-Filtering Router**

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

- **Application level gateways**

  An application-level gateway acts as a relay of application-level traffic. It is also known as proxy server. The user contacts the gateway using a TCP/IP application, such as Telnet or FTP, and the gateway asks the user for the name of the remote host to be

accessed. When the user responds and provides a valid user ID and authentication information, the gateway contacts the application on the remote host and relays TCP segments containing the application data between the two endpoints. If the gateway does not implement the proxy code for a specific application, the service is not supported and cannot be forwarded across the firewall.

Application-level gateways tend to be more secure than packet filters. It is easy to log and audit all incoming traffic at the application level. The main disadvantage of this type of gateway is the additional processing overhead on each connection.

- **Circuit level gateways**

  The Circuit level gateway can be a stand-alone system or it can be a specialized function performed by an application-level gateway for certain applications. A circuit-level gateway does not permit an end-to-end TCP connection, instead the gateway sets up two TCP connections. One connection is set up between itself and a TCP user on an inner host and one between itself and a TCP user on an outside host. Once the two connections are established, the gateway typically relays TCP segments from one connection to the other without examining the contents. The security function consists of determining which connections will be allowed [2].

1.6.2 Honeypot

A honeypot is a trap set to detect, deflect, or in some manner counteract attempts at unauthorized use of information systems. Generally it consists of a computer, data or a network site that appears to be part of a network, but is actually isolated and monitored, and which seems to contain information or a resource of value to attackers. A honeypot works by fooling attackers into believing that it is a legitimate system. The attackers attack the system without knowing that they are being observed. When an attacker attempts to compromise a honeypot, its attack-related information, such as the IP address of the attacker, will be collected. This activity done by the attacker provides valuable information and analysis on attacking techniques, allowing system administrators to trace back to the source of attack if required [6]. In general honeypots can be divided into two categories.

- **Production Honeypots:-**Production honeypots are used to assist an organization in protecting its internal IT infrastructure. These secure the organization by policing its IT environment to identify attacks. These honeypots are useful in catching hackers with criminal intentions. The implementation and deployment of these honeypots are relatively easier than research honeypots because these have less purpose and require

fewer functions. As a result, they also provide less evidence about hacker's attack patterns and motives**.**

- **Research Honeypots:** - Research honeypots are complex. They are designed to collect as much information as possible about the hackers and their activities. Their primary mission is to research the threats organization may face, such as who the attackers are, how they are organized, what kind of tools they use to attack other systems, and where they obtained those tools. While production honeypots are like the police, research honeypots act as their intelligence counterpart and their mission is to collect information about the attacker. The information gathered by research honeypots will help the organization to better understand the hackers attack patterns, motives and how they function [7].

1.6.3 Intrusion Detection System (IDS)

Intrusion Detection System (IDS) helps information systems to deal with attacks. This is accomplished by collecting information from a variety of systems and network sources. The information collected is analyzed for possible security problems. An IDS gathers and analyzes information from various areas within a computer or a network to identify possible security breaches. The intrusions may include attacks both from outside the organization as well as within the organization [8].

- **Advantages of IDS:**
  - **i)** IDS are easier to deploy as it does not affect existing systems or infrastructure.
  - **ii)** Network based IDS sensors can detect many attacks by checking the packet headers for any malicious attack like TCP SYN attack, fragmented packet attack *etc*.
  - **iii)** IDS monitor traffic on a real time. So, network based IDS can detect malicious activity as they occur.
  - **iv)** IDS sensor deployed outside the firewall can detect malicious attacks on resources behind the firewall [8].
- **Disadvantages of IDS**
  - **i)** Not an alternative to strong user identification and authentication mechanism.
  - **ii)** IDS is not a solution to all security concerns.
  - **iii)** Human intervention is required to investigate the attack once it is detected and reported.

iv) False positives occur when IDS incorrectly identifies normal activity as being malicious.

v) False negatives occur when IDS fails to detect the malicious activity [8].

- **Structure and architecture of IDS**

The various components of IDS are shown in Figure 1.1. The IDS components are explained below.

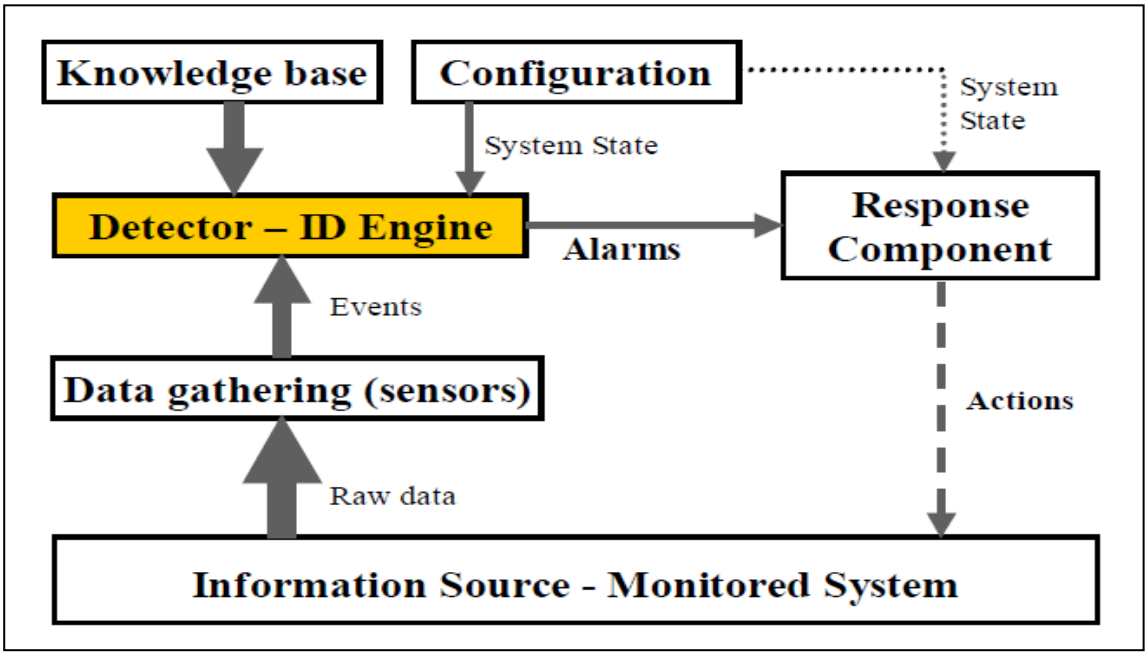


Figure 1.1: Basic architecture of Intrusion Detection System [9]

i) **Data gathering device:** It is responsible for collecting data from the system that is being monitored. It acts like an agent which continuously watches or monitors the network in real time. Examples of input to a sensor are network packets, log files, and system call traces. Sensors collect and forward this information to the analyzer.

ii) **Detector:** It processes the data collected from sensors to identify intrusive activities and uses a system of rules to generate alarms from security events received.

iii) **Knowledge base:** Knowledge base contains information collected by the sensors 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. All the information about the previously detected attack signatures or pattern should be present in the database. When the sensor detects some kind of

malicious activity or signature it matches it with the current database, and report to attack response component.

iv) **Configuration device:** It provides information about the current state of the IDS.

v) **Response component:** It initiates actions when an intrusion is detected. These responses can either be automated or involve human interaction. Response component can either send an alarm or an email notification about the intrusion detected to the administrator depending on the type of configuration [9].

- **Classification of IDS**

  Classification of IDS [9] is discussed in this section. Intrusion Detection Systems can be classified on the basis of data source, method, time of detection, architecture, and reaction which are shown in Figure 1.2.

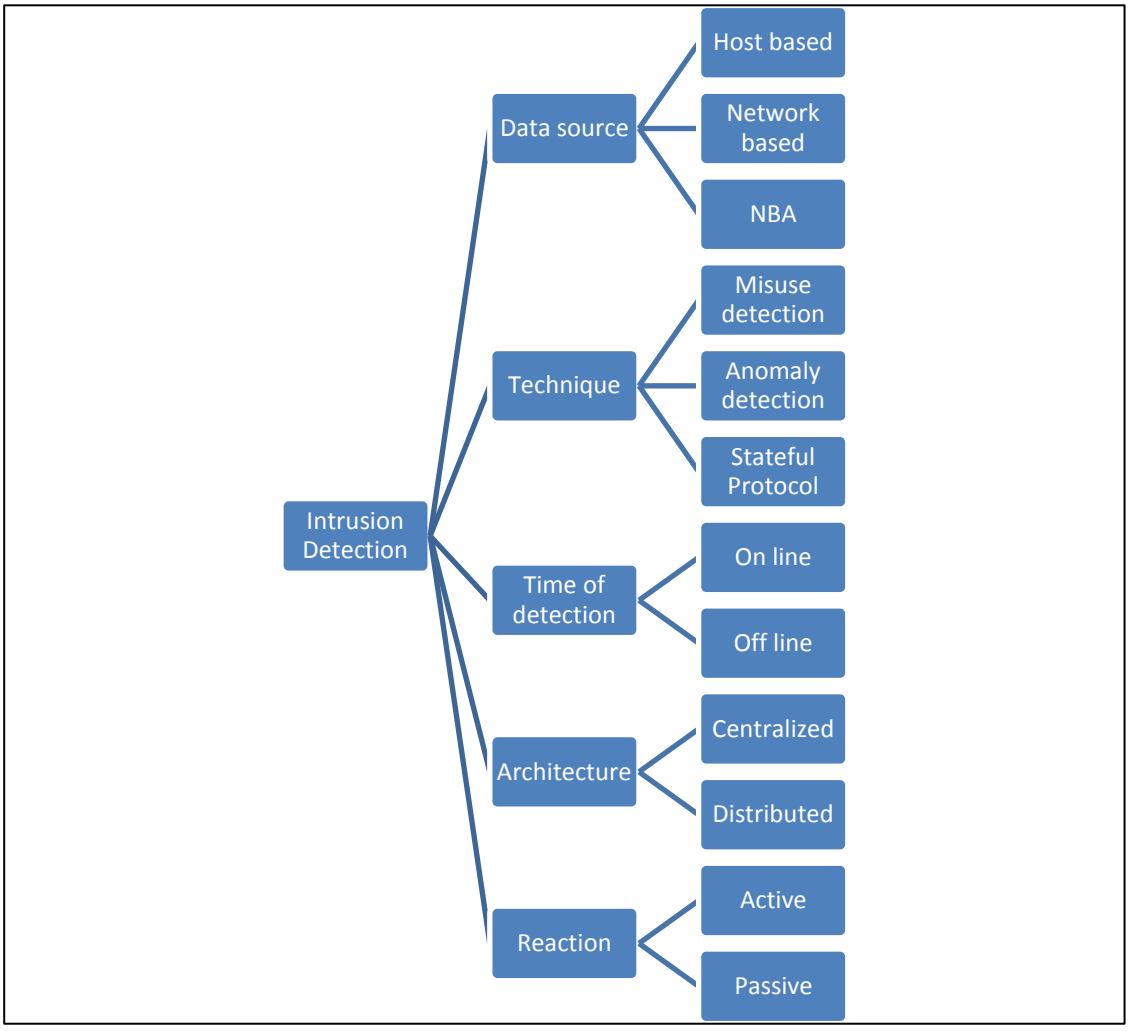


Figure 1.2: Classification of IDS [9]

The Classification of Intrusion Detection Systems listed in the figure above is discussed in next section in detail.

- **Data source**

  An IDS performs 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: Host based intrusion detection systems (HIDS) analyze network traffic and system specific settings such as software calls, local security policy, local log audits, and more. Figure 1.3 shows the architecture of HIDS. A HIDS must be installed on each machine and requires configuration specific to that operating system and software. HIDS takes a snap shot of the existing system files and matches it to the previous snap shot. If the critical system files were modified or deleted, the alert is sent to the administrator to investigate. The example of the HIDS can be seen on the mission critical machines that are not expected to change their configuration.

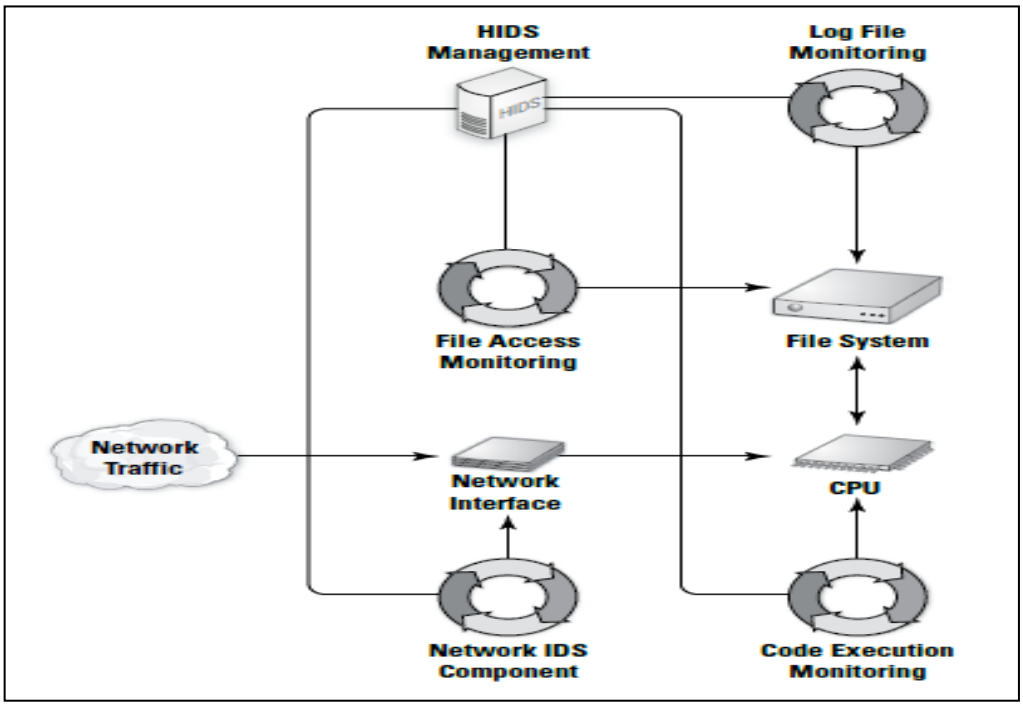


Figure 1.3: Architecture of Host Intrusion Detection System [10]

ii) Network based IDS: A Network Intrusion Detection System (NIDS) is one common type of IDS that analyzes network traffic at all layers of the Open

Systems Interconnection (OSI) model and makes decisions about the purpose of the traffic, analyzing for suspicious activity. Most NIDSs are easy to deploy on a network and can often view traffic from many systems at once. The typical components in an NIDS are as follows and shown in Figure 1.4.

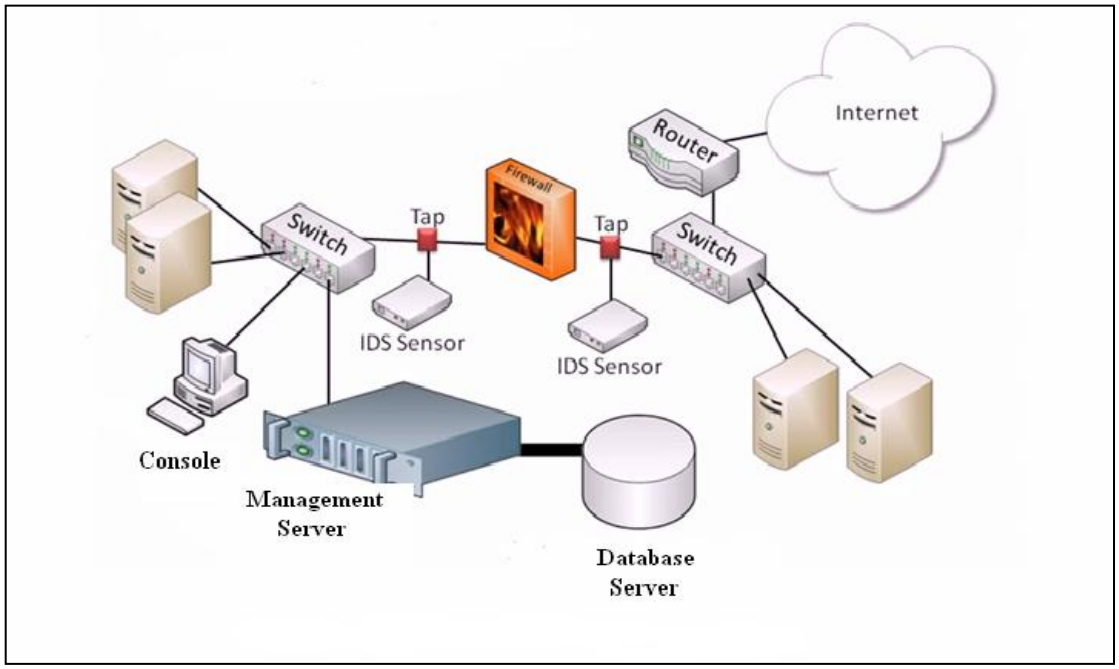


Figure 1.4: Components of Network Intrusion Detection System

a) Sensor: Sensors monitor and analyze the system activity.

b) Management Server: A management server is a centralized device that receives information from the sensors or agents and manages them. Management servers perform analysis on the event information provided by the sensors. Matching event information from multiple sensors or agents, such as finding events triggered by the same IP address, is known as correlation. Management servers are available as both appliance and software based products. Some small IDS deployments do not use any management servers, but most IDS deployments do.

c) Database Server: A database server is a repository for event information recorded by sensors and/or management servers. Many IDS provide support for database servers.

d) Console: A console is a program that provides an interface for the IDS users and administrators. Console software is typically installed onto standard desktop or laptop computers. Some consoles are used for IDS administration purpose only such as configuring sensors and applying software updates, while other consoles

are used strictly for monitoring and analysis. Some IDS consoles provide both administration and monitoring capabilities [10].

iii) **Network Behaviour Anomaly Detection:** Network Behaviour Anomaly Detection (NBAD) is an IDS technology in which the shape or statistics of traffic, not individual packets, determines if the traffic is malicious. NBAD sensors are placed around a network in key places, such as at switches, at demilitarized zones (DMZ), and at locations at which traffic splits to different segments. NBAD requires several sensors to create a good snapshot of a network and requires benchmarking and base lining to determine the nominal amount of a segment's traffic.

- **Technique:** Once required data is collected, an IDS analysis engine processes these data in order to identify intrusive activities. On the basis of method of processing of data, Intrusion Detection Systems are of following types:
  i) **Misuse detection:** It compares known threat signatures to observed events to identify incidents. This is very effective at detecting known threats but largely ineffective at detecting unknown threats. Misuse detection cannot track and understand the state of complex communications, so it cannot detect most attacks that comprise multiple events.
  ii) **Anomaly detection**: It compares definitions of what activity is considered normal against observed events to identify significant deviations. This method uses profiles that are developed by monitoring the characteristics of typical activity over a period of time. The characteristics of current activity ate then compared to thresholds related to the profile. Anomaly detection methods can be very effective at detecting new threats. Common problem with anomaly-based detection is generating many false positives.
  iii) **Stateful protocol:** It compares predetermined profiles of generally accepted definitions of benign protocol activity for each protocol state against observed events to identify deviations. Anomaly detection uses host or network-specific profiles, whereas stateful protocol analysis relies on vendor developed universal profiles that specify how particular protocols should and should not be used. Problem with stateful protocol analysis is that it is often very difficult to develop completely accurate models of protocols [3].

- **Time of detection:** Based on time of detection, IDS can be divided into two groups. Online IDS tries to detect intrusion in real time or near real-time as when the live network traffic comes over network. In Offline IDS first data is collected and then analysis is performed.

- **Architecture:** IDS with centralized architecture detect intrusions that occur in a single monitored system or 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 such as 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:** The response of IDSs to identified attacks may be either passive or active. In the most cases, IDSs have passive response. It simply informs responsible personnel of an event, but no countermeasure is actively applied to stop the attack. The most common method for such notifications is through recording alerts into a file. Alternatively, IDSs can also provide an active response to critical events, such as patching system vulnerability, logging off a user, reconfiguring routers and firewalls, or disconnecting a port. One of the most harmless, but often most productive, active responses is to collect additional information about a suspected attack and to perform damage control.

1.7 Organization of Thesis

Chapter 1 gives a brief introduction about security, security services and mechanisms and attacks threatening security. The various network security solutions that are available in the market are discussed with more emphasis on IDS which is the topic of concern under this thesis.

Chapter 2 provides the survey of various techniques used for detecting intrusions. The classification of IDS based on different criteria is discussed in detail. This chapter also provides information about the systems and tools that are used in the thesis work.

Chapter 3 states the problem statement of the thesis, the objectives and goals to be achieved to carry out the thesis work.

Chapter 4 presents the methodology used to achieve the objectives, the experimental setup, installation and implementation details of the approaches.

Chapter 5 provides the results and main findings of the work carried out.

Chapter 6 draw the conclusion of results and recommendations for future work

Chapter 2

Details of Literature Survey

This chapter is the output of literature survey of the techniques used to detect intrusions. Mainly there are two approaches to Intrusion Detection viz. Misuse detection and Anomaly detection. Both the approaches have advantages and disadvantages. So recently a new approach has come up which combines both the approaches and gives better results. This chapter discusses the classification of IDS based on techniques used to detect intrusions.

2.1 Introduction

There are mainly two appraoaches to detect intrusions. These are Signature detection and Anomaly detection. Figure 2.1 shows the basic working of the techniques used for detecting intrusions. A Signature based IDS analyzes the network traffic and looks for patterns that match a list of known signatures whereas an Anomaly based IDS tries to find suspicious activity on the system. For this purpose, initially the Anomaly based IDS need to be trained in order to get an idea about what is considered as normal. During the testing phase, the system will inform about any suspicious activity based on the training. The next section covers these techniques and their sub categories in detail.

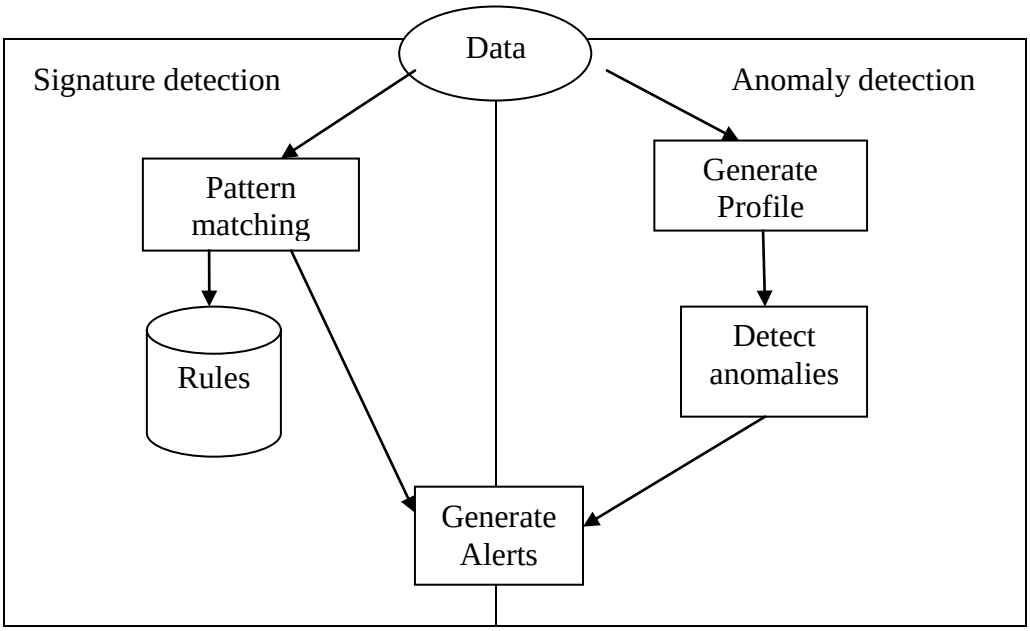


Figure 2.1: Working of signature and anomaly detection [11]

2.2 Techniques used for Detecting Intrusions

The intrusions are detected either by comparing what the IDS is analyzing to a list of signatures or by detecting the abnormal behaviour. The various techniques for detecting the intrusions are discussed in this section. Figure 2.2 shows the intrusion detection techniques and their sub categories.

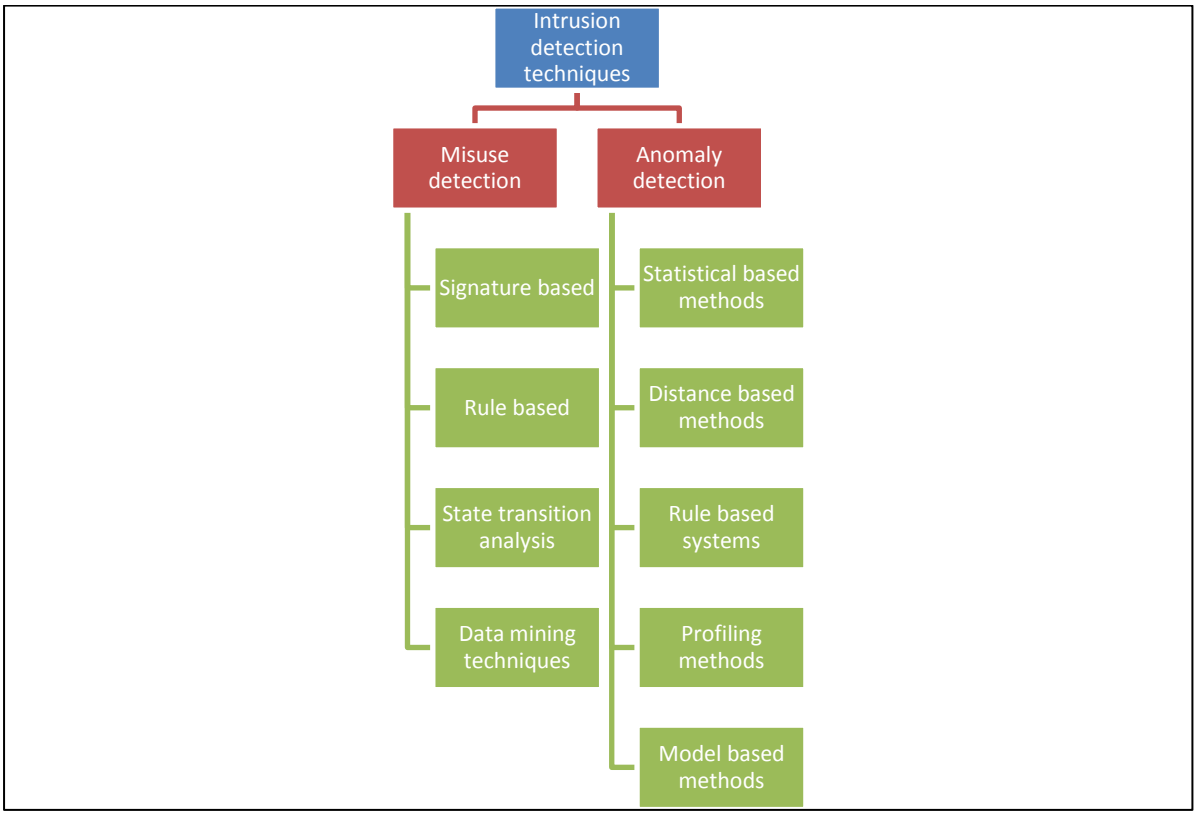


Figure 2.2: Categories of Intrusion detection techniques [9]

2.2.1 Misuse Detection

Misuse detection is based on extensive knowledge of known attacks and system vulnerabilities provided by human experts. The misuse detection approaches look for hackers that attempt to perform these attacks and to exploit known vulnerabilities. Misuse detection is the most common approach used in the current generation of commercial IDS. The misuse detection approaches can be classified into the following four main categories.

i) **Signature based methods:** The signature based approach looks for the signatures of known attacks which exploit weaknesses in system and application software. It uses pattern matching techniques to detect attacks. The pattern in the network traffic is matched to already stored database of signatures. It is useful to detect already known attacks. It fails to detect the new attacks. Signature based IDS technology is widely used

because many attacks have clear and distinct signatures. In signature based IDS, every signature requires an entry in the database, and so a complete database might contain hundreds or even thousands of entries. Each packet is to be compared with all the entries in the database. The various signature based Intrusion Detection Systems are Snort [12] and Bro [13]. These are the widely used open source Intrusion Detection Systems.

ii) Rule based systems: Rule-based systems use a set of "if-then" implication rules to characterize computer attacks. In rule-based IDSs, security events are usually monitored and then converted into the facts and rules that are later used by an inference engine to draw conclusions. Examples of such rule-based IDSs include Shadow [14], P-BEST [15].

iii) State transition analysis: Intrusion detection using state transition analysis requires the construction of a finite state machine, in which states correspond to different IDS states, and transitions characterize certain events that cause IDS states to change. IDS states correspond to different states of the network protocol stacks or to the integrity and validity of current running processes or certain files. Every time when the automation reaches a state that is flagged as a security threat, the intrusion is reported as a sign of malicious attacker activity. This is the technique first proposed in USTAT (Unix State Transition Analysis Tool) [16] and later in NetSTAT (Network-based State Transition Analysis Tool) [17].

iv) Data mining based techniques: In data mining methods for misuse detection, each instance in a data set is labelled as 'normal' or 'intrusive' and a learning algorithm is trained over the labelled data. These techniques are able to automatically retrain intrusion detection models on different input data that include new types of attacks, as long as they have been labelled appropriately. Research in misuse detection has focused mainly on classification of network intrusions using various standard data mining algorithms.

- **Advantages**
  - **i)** Misuse detection systems are very effective at detecting attacks without generating a number of false alarms.
  - **ii)** Misuse detectors can quickly and reliably diagnose the use of a specific attack tool or technique that is used by the attacker. This can help security managers develop corrective measures.
  - **iii)** Misuse detectors can allow system managers to find security problems on their systems.

- **Disadvantages**
  - **i)** Misuse detectors can only detect those attacks that are present in their database. Therefore they must be constantly updated with signatures of new attacks.
  - **ii)** Many Misuse detectors are designed to use tightly defined signatures that prevent them from detecting variants of common attacks.

2.2.2 Anomaly Detection

An IDS that looks at network traffic and detects data that is incorrect, not valid or generally abnormal is called anomaly based detection. This method is useful for detecting unwanted traffic that is not specifically known. The anomaly based approach looks for behaviour or use of computer resources deviating from normal or common behaviour [10]. The available Anomaly based IDSs are PHAD (Packet Header Anomaly Detector) [18], NETAD (Network Traffic Anomaly Detector) [19], ALAD (Application Layer Anomaly Detector) [20] and AnomalyDetection [21, 22]. Anomaly detection is divided into following categories.

i) **Statistical based methods:** Statistical methods monitor the user or system behaviour by measuring certain variables over time such as login and logout time of each session. The basic models keep averages of these variables and detect whether thresholds are exceeded based on the standard deviation of the variable. Some of the first proposed anomaly detection algorithms were integrated in well known IDSs such as NIDES [23] and EMERALD [24].

ii) **Distance based methods:** Distance based approaches attempt to overcome limitations of statistical outlier/anomaly detection approaches and they detect outliers by computing distances among points. MINDS (Minnesota Intrusion Detection System) [25] uses net-flow data to extract useful set of features to be used in anomaly detection.

iii) **Rule based systems:** These systems are used in anomaly detection to characterize normal behaviour of users, networks and/or computer systems by a set of rules. Examples of rule based IDSs include ComputerWatch [26] which employs a typical rule based system that summarizes normal security events and then detects anomalous behaviour as deviations from them. The rule system creates rules to describe proper usage policy, to check the actions of users according to these rules and to flag any action that does not match the described rule patterns.

iv) **Profiling methods:** In profiling methods, profiles of normal behaviour are built for different types of network traffic, users, programs *etc*., and deviations from them are

considered as intrusions. Profiling methods vary greatly ranging from different data mining techniques to various heuristic-based approaches.

v) **Model based methods:** In the model based approaches, anomalies are detected as deviations for the model that represents the normal behaviour. Unsupervised support vector machines [27] attempt to separate the entire training data set from the origin, *i.e.* to find a small region where most of the data lies and label data points in this region as a normal behaviour. In the test phase they detect deviations from learned models as potential intrusions.

- **Advantages**
  - **i)** IDSs based on anomaly detection detect unusual behaviour and thus have the ability to detect new types of attacks.
  - **ii)** Anomaly detectors can produce information that can in turn be used to define signatures for Signature based Detection IDS.
- **Disadvantages**
  - **i)** Anomaly detection approaches usually produce a large number of false alarms.
  - **ii)** Anomaly detection approaches often require extensive training sets of system event records in order to characterize normal behaviour patterns from abnormal behaviour patterns.

2.2.3 Intrusion Detection techniques based on Machine learning method

Existing intrusion detection systems rely heavily on human analysts to differentiate intrusive from normal network traffic. The large and growing amount of data confronts the analysts with an overwhelming task, making the automation of aspects of this task necessary. Whether complete automation is possible or even desirable is debatable. The machine learning techniques can be used which provide decision aids for the analysts and which automatically generate rules to be used for computer network intrusion detection [28].

In machine learning research, a set of training data is used to build or learn for a model in a space of possible models. Given some training data, machine learning is equivalent to searching the hypothesis space or model space for the best possible hypothesis that describes the training data. Evidently a well defined learning problem requires a well formulated problem, a performance metric and a source of training experience [29].

Various machine learning algorithms like Neural Network, Support Vector Machine, Genetic Algorithm, Fuzzy Logic, and Data Mining have been extensively used to detect intrusion activities.

Data mining is also known as knowledge discovery in databases and has attained a great deal of attention in the information industry & in society. Within few years, the availability of large amount of data & its prominent need for extracting such data into useful information is increasing rapidly. The concept of data mining has been around for years. Despite this data mining in intrusion detection is a relatively new concept. Data mining tasks can be classified into two categories namely descriptive mining & predictive mining. The descriptive mining techniques such as clustering, Association, Sequential Pattern discovery, is used to find human interpretable patterns that describe the data. The predictive mining techniques like classification, Regression, and Deviation detection, *etc*., are used to predict unknown or future values of other variables [30].

Applications of Data Mining in Computer Security concentrate heavily on the use of data mining in the area of intrusion detection. The reason for this is twofold. First, the volume of data dealing with both network and host activity is so large that it makes it an ideal candidate for using data mining techniques. Second, intrusion detection is an extremely critical activity. An ideal application in intrusion detection will be to gather sufficient "normal" and "abnormal" audit data for a user or a program, then apply a classification algorithm to learn a classifier that will determine audit data as belonging to the normal class or the abnormal class [31].

Classification problems aim to identify the characteristics that indicate the group to which each case belongs. This pattern can be used both to understand the existing data and to predict how new instances will behave. Data mining creates classification models by examining already classified data and inductively finding a predictive pattern. This already classified data may come from historical database or from an experiment in which a sample of the entire database is tested in the real world and the results are used to create a classifier. A number of data mining algorithms are available that perform summarization of the data, classification of data with respect to a target attribute, deviation detection, and other forms of data characterization and interpretation. In data mining tasks, classification and prediction is among the popular task for knowledge discovery and future plan. The classification process is known as supervised learning, where the class level or classification target is already known. There are many techniques used for classification in data mining such as Decision Tree, Bayesian, Fuzzy Logic, JRip and Support Vector Machine (SVM). The JRip classification algorithm is easy to understand as the derived rules have a very straightforward interpretation. Due to these reasons, this study uses JRip classification algorithm to handle the data.

Rip (RIPPER) is one of the basic and most popular algorithms. Classes are examined in increasing size and an initial set of rules for the class is generated using incremental reduced error. JRip (RIPPER) proceeds by treating all the examples of a particular judgment in the training data as a class, and finding a set of rules that cover all the members of that class. Thereafter it proceeds to the next class and does the same, repeating this until all classes have been covered [32].

- **Advantages**
  - **i)** This technique is complementary to signature analysis.
  - **ii)** It is efficient for detection of unknown attacks.
  - **iii)** It provides high detection accuracy with low false positives.
- **Disadvantages**
  - **i)** It is possible for a company to collect millions of records per day which need to be analyzed for malicious activity. Millions of records collected per day are to be analyzed, so data mining will become quite computationally expensive and its complexity will increase dramatically.
  - **ii)** For some processing power or memory is not always cheap or available. On the other hand analyzing only samples of data rather than all the data could lead to false conclusions.
  - **iii)** Another obstacle will be tailoring data mining algorithms and processes to fit intrusion detection.

2.2.4 Survey of Existing Hybrid Intrusion Detection Systems

The above mentioned approaches have both advantages and disadvantages. None of the approach is better than the other. Each approach has some limitations. No approach is capable of detecting all the attacks. So many authors have proposed IDS based on a hybrid approach which consists of combination of techniques mentioned above. Hybrid based IDS has been discussed by the following authors in their research work.

Lee *et al.* (1999) [33] presents a data mining framework for adaptively building Intrusion Detection models. The auditing programs are utilized to extract an extensive set of features that describe each network connection or host session. Data mining algorithms are applied to learn rules so as to capture the behaviour of intrusions and normal activities. These rules can then be used for misuse detection and anomaly detection.

Lee *et al.* (2007) [34] suggests a hybrid approach for real time Intrusion Detection System. The proposed approach uses Random Forest (RF) for feature selection and Minimax Probability Machine (MPM) for intrusion detection.

Hwang *et al.* (2007) [35] discusses the design principles and evaluation results of a new experimental hybrid intrusion detection system (HIDS). This hybrid system combines the advantages of low false-positive rate of signature-based intrusion detection system and the ability of anomaly detection system to detect unknown attacks. An Anomaly detection system is built that detects anomalies that are not detected by signature based Snort or Bro systems. A weighted signature generation scheme is developed to integrate ADS with Snort by extracting signatures from anomalies detected. Signatures from the output of ADS are extracted and added to the Snort signature database for intrusion detection. HIDS scheme is tested over real-life Internet trace data mixed with 10 days of Massachusetts Institute of Technology/Lincoln Laboratory (MIT/LL) attack data set. The experimental results show a 60 percent detection rate of the HIDS, compared with 30 percent and 22 percent in using the Snort and Bro systems, respectively.

Díaz-Verdejo *et al.* (2007) [36] proposes a modified version of Snort to operate as hybrid detector. This version can be used during the training phase of the system. The system may also be adjusted to operate only as an anomaly based detector or Hybrid detector.

Zhang *et al.* (2008) [37] describes new systematic framework that applies random forests, commonly known data mining program, in misuse, anomaly, and hybrid-network-based IDSs. In signature detection, patterns of intrusions are built automatically over training data by random forest algorithm. After that, intrusions are detected by matching network activities against the patterns. In anomaly detection, the outlier detection mechanism of the random forests algorithm is used to detect the intrusions. The hybrid detection system improves the detection performance by combining the advantages of the misuse and anomaly detection.

Aydin *et al.* (2009) [38] combines the anomaly detection approach and the signature detection approach to form hybrid IDS. The scheme uses packet header anomaly detection (PHAD) and network traffic anomaly detection (NETAD), which are anomaly-based IDSs with the misuse-based IDS Snort, which is an open-source project. The hybrid IDS obtained is tested on the MIT Lincoln Laboratories network traffic data (IDEVAL). Evaluation shows that the hybrid IDS is a more powerful system than the individual schemes.

Gómez *et al.* (2009) [39] adds a new anomaly preprocessor to Snort to extend the functionality of Snort IDS. This hybrid IDS is named as H Snort. It has been tested with the

DARPA dataset. Results obtained have shown that the performance of the IDS is affected by database or the number of elements used to model the normal network behaviour.

Zhao *et al.* (2010) [40] proposed a hybrid IDS which combines network and host IDS with anomaly and misuse detection mode. It utilizes auditing programs to extract an extensive set of features that describe each network connection or host session. Data mining programs are applied to learn rules that accurately capture the behaviour of normal activities and attacks.

Yang *et al.* (2010) [41] suggests a hybrid intrusion detection system using both misuse detection and anomaly detection techniques. In misuse detection model, the protocol type of the data instance is identified by the protocol selector and then misuse detection engine picks the most efficient decision tree algorithm. In anomaly detection model, all three decision tree algorithms are tested on data instances and the most efficient one is derived. The performance of HIDS is tested on a model based on Generalized Stochastic Petri Nets. The results show that this system provides a high detection rate with low false-positive rate and false-negative rate.

Amza *et al.* (2011) [42] proves in his paper that using a hybrid intrusion detection system, the detection capabilities can be increased. In this paper a pattern matching engine is combined with a neural network detection component to detect anomalies in the network traffic. This approach is able to efficiently detect known attacks as well as unknown attacks. In this approach the two detection techniques are run in parallel. To minimize the number of notifications which will be sent to the network administrator a filter is added.

Hussein *et al.* (2012) [43] proposed a hybrid IDS by integrating signature based system Snort with anomaly based Naive Bayes to enhance system security to detect attacks. They used Knowledge Discovery Data Mining (KDD) CUP 99 dataset and Waikato Environment for Knowledge Analysis (WEKA) program for testing the proposed hybrid IDS.

Qazanfari *et al.* (2012) [44] used Support Vector Machine (SVM) and Multi Layer Perceptron (MLP) as anomaly detection approaches to develop a hybrid intrusion detection approach.

Jain and Sardana (2012) [45] focus on a novel robust hybrid approach which increases the accuracy the system. The results show increase in detection rate by 32.78 % in comparison with the existing approach that uses signature with anomaly detection which is best among existing approach. The false alarm rate is also reduced by 33.3%.

Nadiammai and Hemalatha (2012) [46] develops a hybrid IDS by combining the anomaly based detection approaches like Packet Header Anomaly Detector (PHAD) [18],Network Traffic Anomaly Detector (NETAD) [19], Application Layer Anomaly Detection (ALAD)

[20]. The hybrid IDS obtained is evaluated using the KDD Cup 99 traffic data and real time data. The performance of hybrid IDS is measured by comparing the number of attacks detected by misuse based IDS with the hybrid IDS. The results show that the hybrid IDS with ALAD and LERAD performs better by detecting 149 attacks out of 180 (83%) attacks after training on one week attack free traffic data.

2.3 Snort: A Signature based Intrusion Detection System

Snort [12] is a small, lightweight Intrusion Detection System written by Martin Roesch. Snort is an open source Intrusion Detection System, which may also be configured as an Intrusion Prevention system for monitoring and prevention of security attacks on networks. It is a cross-platform 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 report an attack. Snort performs protocol analysis and content matching to detect intrusions. 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.

2.3.1 Snort components

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. The Snort components are shown in Figure 2.3 and are explained below.

i) **Packet capture block:** This module is used for capturing the packets. The packets are captured from network interfaces and passed to decoder module for further processing.

ii) **Decoder:** It is the module responsible to perform the syntax analysis at MAC, IP, TCP/UDP layer of the IP packet.

iii) **Preprocessors:** It is the block where multiple preprocessors can be loaded at boot time to analyze protocols of layers above the TCP/UDP with custom made C/C++ programs.

iv) **Detection engine:** Detection engine finds out if intrusion activity is present in packet or not. Snort rules are used for this purpose. If any rule is matched, appropriate action is taken. Action may include generating the alerts.

v) 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 is also generated by

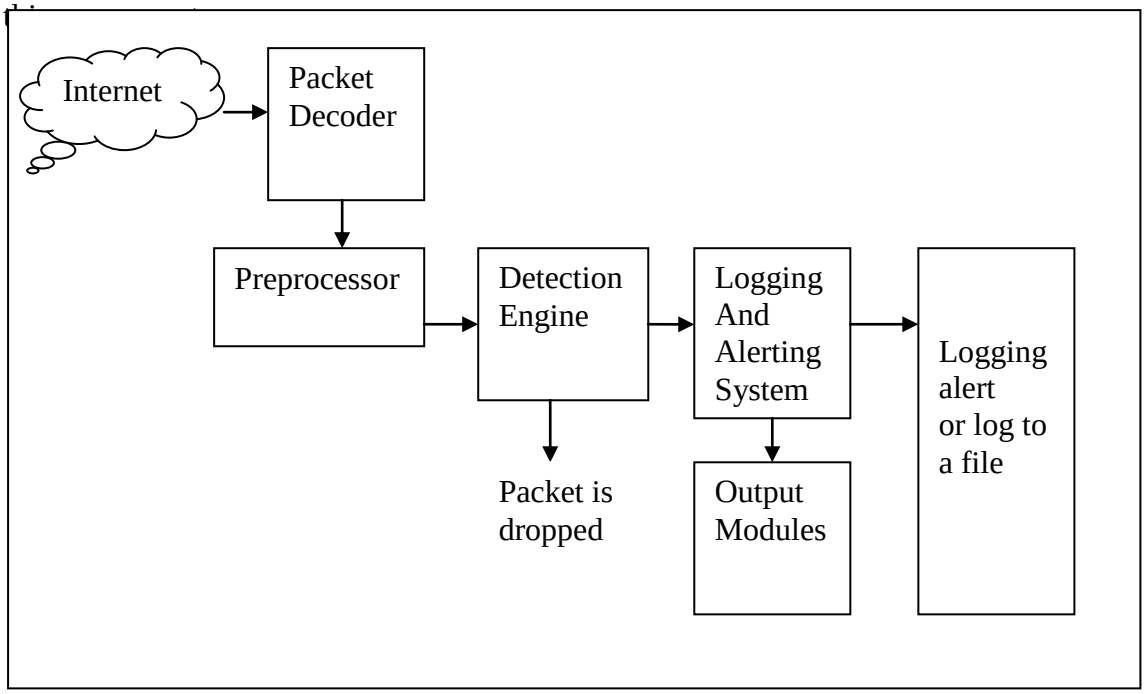


Figure 2.3: Components of Snort [46]

2.3.2 Features of Snort

The basic feature of Snort include packet capturing, logging, performing real time analysis, content searching and matching and detecting attacks which are explained in detail as follows:

i) **Packet capturing and logging:** Snort captures the packets using Libpcap or Winpcap to capture the packets coming on an interface.

ii) **Real time traffic analysis:** Snort is capable of performing analysis of packets coming on the network in real time. It is also capable of performing analysis on captured packets which can be captured using various tools like Wireshark.

iii) **Content searching/matching:** Snort performs the content searching of the packet's content against the rules in the rules file.

iv) **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.

2.3.3 Writing snort rules

Snort uses a simple, lightweight rules description language that is flexible and quite powerful. This section explains the instructions followed while writing Snort rules [47]. Snort rules have two parts *i.e.* Rule header and Rule options.

- **Rule header**

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

  i) Rule Actions: The rule action tells Snort what to do when it finds a packet that matches the rule criteria. There are five available default actions in Snort.

  a) alert: Generate an alert using the selected alert method, and then log the packet

  b) log: Log the packet

  c) pass: Ignore the packet

  d) activate: Alert and then turn on another dynamic rule

  e) dynamic: Remain idle until activated by an activate rule, then act as a log rule

  f) drop: Make iptables, drop the packet and log the packet

  g) 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.

  h) sdrop: Make iptables drop the packet but does not log it.

  ii) Protocol: The next field in a rule is the protocol. There are four protocols that Snort analyzes for suspicious behaviour that are TCP, UDP, ICMP and IP.

  iii) 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.

  iv) 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.

v) **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.

- **Rule options**

  Rule options [47] form the most important part 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.

  i) **General rule options:** These options provide information about the rule but do not have any affect during detection**.**

  ii) **Payload detection rule options:** These options all look for data inside the packet payload and can be inter-related.

  iii) **Non-Payload detection rule options:** These options look for non-payload data.

  iv) **Post-detection rule options:** These options are rule specific triggers that happen after a rule has fired.

2.3.4 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.

i) **Signature detection rule:** Snort handles its signatures based detection with the rules created for it. The signatures allows a rule to specify a byte string to match against anywhere in the packet including the payload data. Example of the rule is shown in (2.1).

alert tcp any any -> any 80 (msg: "CodeRedII root exe"; flags: A+; content: "root.exe"; depth:624; classtype: attempted-admin;) ...(2.1)

The rule looks for the "CodeRedII" worm in the packet and displays the message "CodeRedII root exe" if the worm is detected.

ii) Protocol detection rule: Snort is provided with rule set for detecting protocols and suspicious behaviour of these protocols in use. Snort rules have the protocol field that allows the detection of the protocol. Example is shown in (2.2).

alert tcp any any -> 192.168.1.0/24 any (flags: SF; msg: "SYNC-FIN packet detected";) (2.2)

The rule detects any scan attempt using SYN-FIN TCP packets. The flags keyword is used to find out which flag bits are set inside the TCP header of a packet.

iii) Anomaly detection rule: 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. Example is shown in (2.3).

alert ip any any -> 192.168.1.0/24 any (dsize: > 6000; msg: "Large size IP packet detected";) ...(2.3)

The rule generates an alert if the data size of an IP packet is larger than 6000 bytes [48].

2.3.5 Operation modes of Snort

Snort runs in different operational modes which are as follows:

- **Sniffer Mode:** This mode 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:** This 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 specified directory.

  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):** It is 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 –c snort.conf

where snort.conf is the name of Snort configuration file. This will apply the rules configured in the snort.conf file to each packet to decide if an action based upon the rule type in the file should be taken.

- **Inline Mode:** It 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. This is the mode used when Snort is to act as an IPS [47].

2.3.6 Preprocessors

Preprocessors [47] allow the functionality of Snort to be extended by allowing users and programmers to drop modular plugins into Snort easily. Preprocessor code is run before the detection engine is called, but after the packet has been decoded. The packet can be modified or analyzed in an out-of-band manner using this mechanism. Preprocessors are loaded and configured using the preprocessor keyword. The format of the preprocessor directive in the Snort config file is:

preprocessor <name>: <options>

2.3.7 Output Modules

Output modules [47] of Snort provide more flexiblity in the formatting and presentation of output to its users. The output modules are run when the alert or logging subsystems of Snort are called, after the preprocessors and detection engine. The format of the directives in the config file is very similar to that of the preprocessors. Multiple output plugins may be specified in the Snort configuration file. When multiple plugins of the same type (log, alert) are specified, they are stacked and called in sequence when an event occurs. As with the standard logging and alerting systems, output plugins send their data to /var/log/snort by default.

Output modules are loaded at runtime by specifying the output keyword in the config file as follows:

output <name>: <options>

Name specifies the name of the output plugin.

This thesis work required the use of two output plugins which are discussed below.

i) CSV: The csv output plugin allows alert data to be written in a format easily importable to a database. The output fields and their order may be customized. The format is

output alert_csv: [<filename> [<format> [<limit>]]]

• filename: the name of the log file. The default name is <logdir>/alert.csv. The name may include an absolute or relative path.

• format: The list of formatting options is below. If the formatting option is default, the output is in the order of the formatting options listed.

Timestamp, sig generator, sig id, sig rev, msg, proto, src, srcport, dst, dstport, ethsrc, ethdst, ethlen, tcpflags, tcpseq, tcpack, tcplen, tcpwindow, ttl, tos, id, dgmlen, iplen, icmptype, icmpcode, icmpid, icmpseq

• limit: an optional limit on file size which defaults to 128 MB. The minimum is 1 KB.

ii) Unified2: The unified output plugin is designed to be the fastest possible method of logging Snort events. The unified output plugin logs events in binary format, allowing other programs to handle complex logging mechanisms that would otherwise diminish the performance of Snort. The unified2 output plugin is a replacement for the unified output plugin. It has the same performance characteristics, but a slightly different logging format. Unified2 can work in one of three modes, packet logging, alert logging, or true unified logging. Packet logging includes a capture of the entire packet and is specified with log unified2. Likewise, alert logging will only log events and is specified with alert unified2. To include both logging styles in a single, unified file, simply specify unified2. The format is

output unified2: filename <base file name> [, <limit <size in MB>]

filename specifies the name of the file and size limit of the file can be specified in MB.

2.4 AnomalyDetection

AnomalyDetection [21] is a preprocessor designed to log and analyze information about network traffic and detect possibly network traffic anomalies. AnomalyDetection analyses the TCP traffic, ICMP traffic, UDP traffic, speed of sending and receiving of packets. It works on 25 parameters of network traffic. These parameters are as follows:

- Number of TCP packets
- Number of outgoing TCP packets
- Number of incoming TCP packet
- Number of TCP packets in LAN
- Number of UDP datagrams
- Number of outgoing UDP datagrams

- Number of incoming UDP datagrams
- Number of UDP datagrams in LAN
- Number of ICMP messages
- Number of outgoing ICMP messages
- Number of incoming ICMP messages
- Number of ICMP messages in LAN
- Number of TCP packets with set SYN and ACK flags
- Number of outgoing TCP packet send on port 80
- Number of incoming TCP packet from port 80
- Number of outgoing UDP datagrams send on port 53
- Number of incoming UDP datagrams from port 53
- Outgoing IP bit rate
- Incoming IP bit rate
- Outgoing TCP port 80 bit rate
- Incoming TCP port 80 bit rate
- Outgoing UDP bit rate
- Incoming UDP bit rate
- Outgoing UDP port 53 bit rate
- Incoming UDP port 53 bit rate

2.4.1 Structure of the system

The Figure 2.4 shows the schematic diagram of the system. AnomalyDetection is added to Snort as preprocessor. A log file is build by running the system in NIDS mode. This is done to log the kind of traffic that comes over the network. The profile generator component reads this log file to generate a profile file. The profile file is read by the preprocessor to detect anomalies [49].

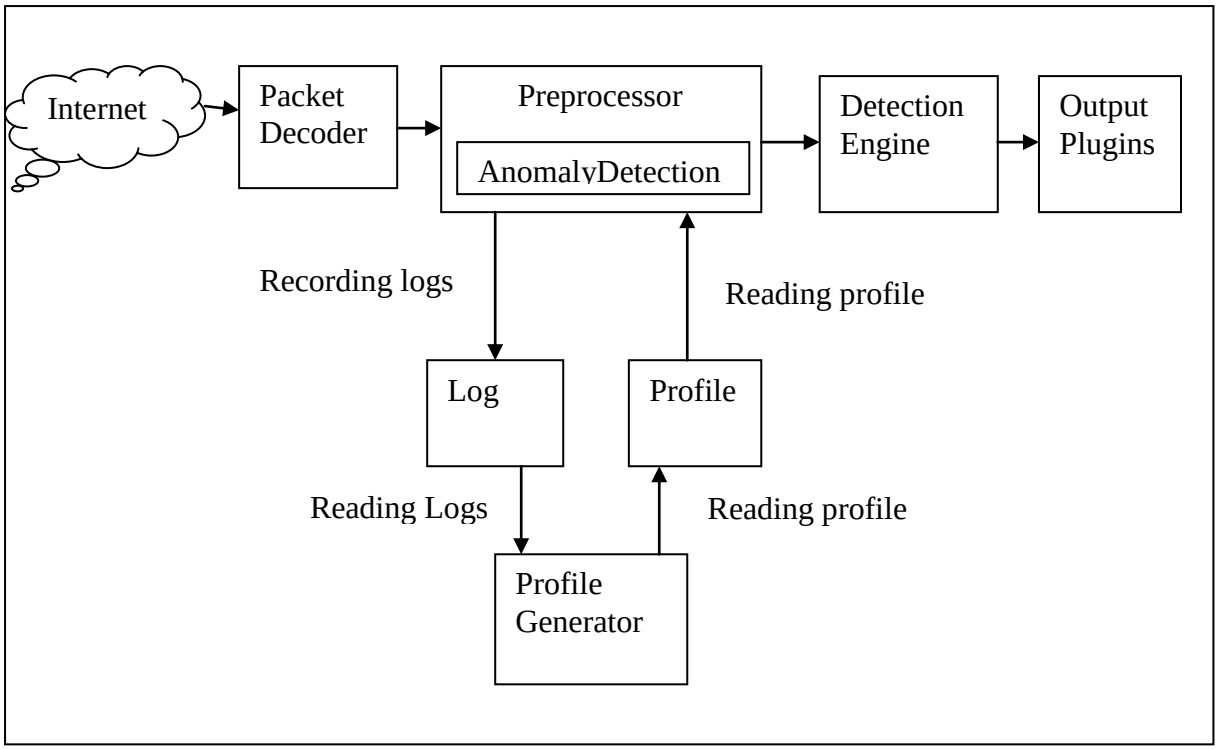


Figure 2.4: Schematic diagram of the system [49]

2.4.2 Working and Components of Anomalydetection

The Figure 2.5 shows the various components of AnomalyDetection and the working of the system. The components are discussed in the following section.

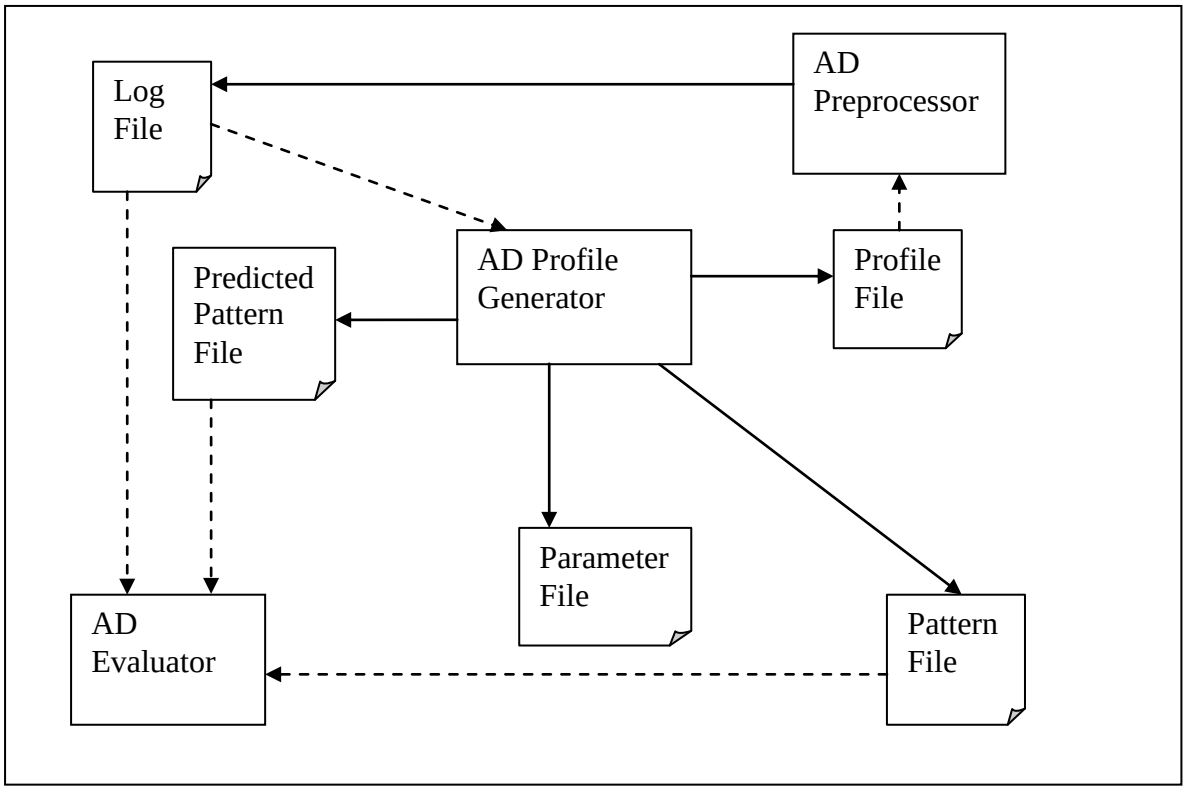


Figure 2.5: Components and Working of AnomalyDetection [50]

The components of AnomalyDetection are as follows:

- **AD Profile Generator**: The profile can be generated manually or by a Profile Generator using appropriate model. The profile is generated based on the log file. First a log file is created based on the network. The log file will contain a profile of the network traffics characteristics. The profile generator of AD is designed based on R environment [51].
  The Profile generator can build profiles based on four methods:
  i) Moving average
  ii) Naive method
  iii) Autoregressive time series model
  iv) Holt-Winters model
- **Preprocessor**: The function of the preprocessor is generating alerts. Preprocessor reads a predicted pattern of the network traffic of all parameters from the profile file and generates alert when the current value exceeds minimum to maximum range for the current moment from the profile file. The moment is given by day of the week, hour, minute and second corresponding to the intervals from the log file.
- **Evaluator**: It is designed to compare the statistic Mean absolute error for two files. The Evaluator can be used either for checking fit between the model and historical data or between the predicted and real values [50].

The working of AnomalyDetection is as follows:

AnomalyDetection writes to a log file which contains information about the traffic like time, day and the 25 parameters discussed above. Profile generator reads the log file and then prepares a profile for each hour of the day which contains the average and standard deviation value for the parameters. This result in creation of network profile created for the whole week. The profile file specifies average and standard deviation for the above specified parameters. The profile file contains data for full one week. There are total 168 lines in the profile file, containing values for each hour of the day for one week.

This profile file is used by preprocessor to generate alerts. Two types of alerts can be generated after analyzing the traffic on the following basis.

- Traffic is too high
- Traffic is too low

Preprocessor reads the current system time and day and then finds the line which contains data about traffic on that day and that hour. An alert is generated according to the following formula.

$W \notin (w-2\sigma; w+2\sigma)$

Where W is the current value of the traffic and σ is value of standard deviation read from profile file.

If the current value of the traffic is greater than adding standard deviation multiplied by two to average value read from the profile file, an alert is generated for too high traffic.

If the current value of the traffic is less than subtracting standard deviation multiplied by two from average value read from the profile file, an alert is generated for too low traffic [49].

2.5 DARPA/MIT Lincoln Laboratory Intrusion detection Evaluation Dataset

This section discusses the DARPA/MIT Lincoln Laboratory Intrusion detection Evaluation dataset [52] which has been used in the thesis for experimentation purpose.

2.5.1 Introduction

The DARPA 1999 intrusion detection off-line evaluation is used for evaluation of the capabilities of current intrusion detection systems. The dataset is of interest to all researchers who are working on the general problem of workstation, or host-based, and network intrusion detection. The dataset eliminates the security and privacy concerns that are caused when the data is made publicly available. It also provides data types that are used by the majority of intrusion detection systems. The objective of dataset was to test the Intrusion detection systems on a wide range of attacks. Another objective was to provide off-line data to encourage development of new systems and algorithms by publishing a standard benchmark so that researchers could compare systems and replicate results.

2.5.2 Description of dataset

The set consists of network traffic (tcpdump files) collected at two points, BSM logs, audit logs, and file system dumps from a simulated local network of an imaginary air force base over a 5 week period. The simulated system consists of four real victim machines running SunOS, Solaris, Linux, and Windows NT, a Cisco router, and a simulation of a local network with hundreds of other hosts and thousands of users and an Internet connection without a firewall. The evaluation data set is collected from the four victim machines and from two network sniffers, an inside sniffer between the router and the victims, and an outside sniffer between the router and the Internet. Attacks may originate from either the Internet, from compromised hosts on the local network, or from attackers who have physical access to the

victims or the local network. Most attacks are against the four main victim machines, but some are against the network, the Cisco router, or against simulated hosts on the local network.

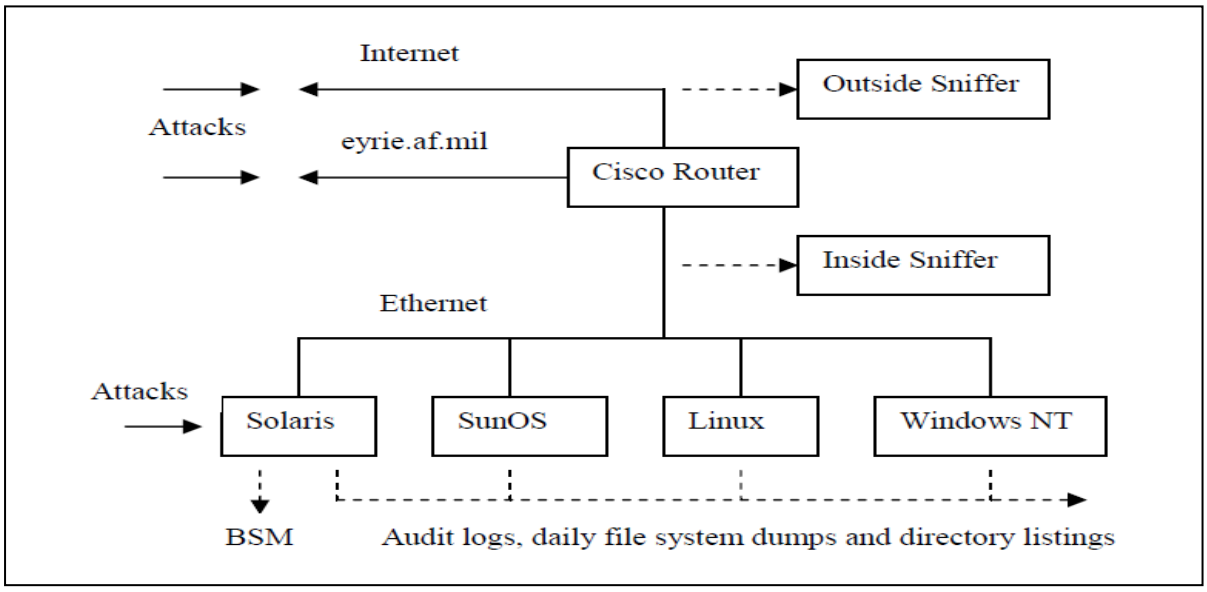


Figure 2.6: The 1999 DARPA/Lincoln Laboratory IDS Evaluation Test Configuration [53]

The data set consists of weeks one, two and three of training data and weeks four and five of test data. In training data, the weeks one and three consist of normal traffic and week two consists of labelled attacks. Two weeks of network based attacks in the midst of normal background data. The fourth and fifth weeks of data are the Test Data used in the 1999 Evaluation from 29/03/1999 to 10/04/1999. There are 201 instances of about 56 types of attacks distributed throughout these two weeks [52].

Kendall (1999) [54] describes a taxonomy of attacks with respect to the DARPA Intrusion Detection Evaluation dataset, grouping them into four major categories.

- Probes: testing a potential target to gather information. These are usually harmless (and common) unless a vulnerability is discovered and later exploited.
- Denial of service (DOS): attacks which prevent normal operation, such as causing the target host or server to crash, or blocking network traffic.
- Remote to local (R2L): attacks in which an unauthorized user is able to bypass normal authentication and execute commands on the target.
- User to root (U2R): attacks in which a user with login access is able to bypass normal authentication to gain the privileges of another user, usually root.

With the introduction of new applications, newer kinds of attacks are targeting the systems. The DARPA IDS Evaluation dataset contains old attacks. So evaluating IDS on the dataset

may not give best results. Although there are many drawbacks of using DARPA IDS Evaluation dataset but still there is no good data set other than DARPA data set for IDS evaluation. The dataset is in tcpdump format which is required in this thesis work. The next section discusses the criticism and advantages of the dataset.

2.5.3 Facts in support of the DARPA IDS Evaluation data set

- The non availability of any other data set that includes the complete network traffic was probably the initial reason to make use of the DARPA data set for evaluation by a researcher in IDS.
- Using real network traffic for evaluation purpose is difficult. The difficulty is lack of the information regarding the status of the traffic. The DARPA IDS Evaluation dataset provides details about the dataset like the no. of attacks in the data, category of attack *etc*. Hence detection rate of IDS can be calculated and is useful for comparison purpose.
- Mahoney and Chan (2003) [55] points out that if an advanced IDS could not perform well on the DARPA data set, it could also not perform acceptably on realistic data. Hence it is better to use DARPA dataset and check whether IDS detects all the attacks of the DARPA data set. Brugger and Chow (2005) [56] used DARPA IDS evaluation dataset. According to them, it may still be useful for a first-order IDS evaluation.

2.5.4 Criticisms against the DARPA IDS Evaluation data set

- The popular critiques to the DARPA IDS Evaluation data set are by Mc Hugh (2000) [57]. He criticizes the procedures used in building the data set and in performing the evaluation. In his critique of DARPA evaluation and if real background traffic was used, the false positive rate would be much higher.
- All the above criticisms have been well researched comments and these works have made it clear that there remain several issues unsolved in design and modelling of the resultant data set.
- The DARPA data set has the drawback that it was not recorded on a network connected to the Internet. Internet traffic usually contains a fairly large amount of anomalous traffic that is not caused by any malicious behaviour. Hence the DARPA data set being recorded in a network isolated from the Internet might not include these types of anomalies.

However, in the lack of better benchmarks, vast amount of the research is based on the experiments performed on the DARPA data set. Even with all the criticisms, the DARPA data set is still rigorously used by the research community for evaluation of IDSs.

2.6 Introduction to Wireshark

Wireshark [58] acts like a packet sniffer. Packet Sniffer is used to observe the messages exchanged between executing protocol entities. A packet sniffer captures messages being sent or received from or by the computer. It stores and displays the contents of the various protocol fields in these captured messages. A network packet analyzer will try to capture network packets and display that packet data as detailed as possible. As a sniffer, Wireshark is a very useful and powerful network packet analyzer. In the past, these tools were either very expensive, proprietary, or both. Wireshark is a free network protocol analyzer that runs on Windows, Linux/Unix and Mac computers. It is an ideal packet analyzer.

Wireshark can open the captured tcpdump files for analysis. These files can be exported into various formats like CSV (Comma Separated Values), Plain text file *etc*. The tcpdump files of the DARPA IDS Evaluation dataset are converted into CSV file for the work of the thesis.

2.7 Introduction to WEKA

WEKA [59] stands for Waikato Environment for Knowledge Analysis .WEKA toolkit is a widely used for machine learning and data mining that was originally developed at the University of Waikato in New Zealand. It contains large collection of state-of-the-art machine learning and data mining algorithms written in Java. It is a collection of machine learning algorithms for data mining tasks. The algorithms are applied directly to a dataset. WEKA implements algorithms for data preprocessing, classification, regression, clustering and association rules. It also includes tools for visualization. WEKA is open source software issued under General Public License. The data mining software selected for the thesis work is WEKA so as to find interesting patterns in the data, extract rules and perform analysis. The data file normally used by WEKA is in ARFF file format, which consists of special tags to indicate different things in the data file. However csv format and some other formats can also be used. The format contains attribute names, attribute types, attribute values and the data. The main interface in WEKA is the Explorer. It has a set of panels, each of which can be used to perform a certain task.

2.8 Summary of the chapter

In this chapter, the techniques to detect intrusions are discussed. There are mainly two methods to detect intrusions, anomaly detection and misuse detection. These techniques are further classified into various categories which are discussed with examples. The new technique, machine learning applied to intrusion detection is explained. The survey of the existing hybrid intrusion detection methods is presented followed by the systems and tools that are used in the thesis work. These are Snort, AnomalyDetection (AD), Wireshark and WEKA. The details of the DARPA IDS Evaluation dataset.

Chapter 3

Problem Statement

The current scenario of network security is very complicated. The statistics show that cyber crime has risen over the years. Enterprises are targeted by hackers either to steal the confidential information or just harm the organization by disrupting their services. Network security has really become important over the recent years. The organizations need to protect their network from attacks which occur from outside as well as within the organization.

For the purpose of providing the solution to the problem of cyber attacks, many security solutions have been developed like antivirus, firewall, Intrusion Detection Systems, honeypots etc. The topic of concern in this thesis is Intrusion Detection System. There are many open source and commercial IDS available in the market. The widely used open source IDS is Snort.

The basic techniques or approaches to detect intrusions are anomaly detection and signature detection. Anomaly detection approach can detect new attacks by detecting the intrusions based on behaviour deviating from normal. On the other hand, signature detection approach detect intrusions by comparing the network traffic to a known database of signatures and reports a very low false alarm rate.

The objectives of the thesis include the following:

- To test a hybrid IDS approach which is a combination of anomaly based IDS and signature based IDS.
- To make a testbed for testing the above model.
- To test and validate the signature based IDS on DARPA IDS Evaluation dataset.
- To test and validate the anomaly based and signature based IDS system on DARPA IDS Evaluation dataset.
- To compare the performance of signature based IDS with anomaly + signature based IDS on the basis of following parameters:
  - **i)** Number of distinct alerts generated date wise
  - **ii)** Number of alerts generated protocol wise
  - **iii)** Number of attacks detected or Detection rate
- To analyze the results using a machine learning tool.

Chapter 4

Implementation and Experimental Results

The solution to the problem discussed in the previous chapter is to develop an Intrusion Detection System which is based on the combined approach of anomaly and signature detection. This chapter provides the implementation details of the approach. The installation steps of signature and anomaly based IDS on Ubuntu are explained. The methodology used to implement the objectives and the experiments performed are explained in this chapter.

4.1 Experimental Setup

The experimental setup of the work done is shown in Figure 4.1. Ubuntu and Backtrack operating systems, the flavours of Linux OS are used. Using virtualization, Ubuntu OS and Backtrack OS are installed as virtual machine on Windows OS with the help of Oracle VM VirtualBox. Signature detection is implemented using Snort IDS and anomaly detection is implemented using AnomalyDetection. Installation of Snort and AnomalyDetection is done on Ubuntu OS. Snort is already installed on Backtrack OS. Installation of MySQL is also done on both machines.

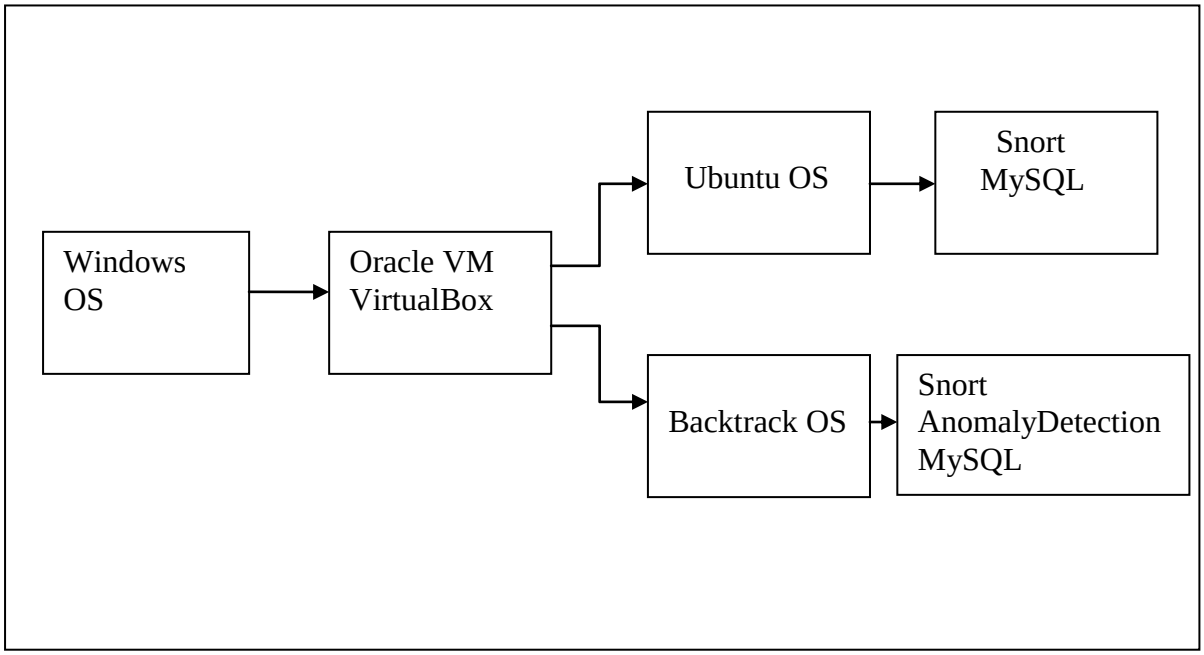


Figure 4.1: Experimental Setup

4.2 Methodology

This section presents the methodology used to carry out the thesis work. Figure 4.2 describes the schematic diagram of the system combining the approach of anomaly and signature detection. The system captures the packets from network traffic. At the first level, any deviation from the normal behaviour can be easily detected by anomaly detector by making use of profile. At the second level, signature detection is used which detects the known attacks from incoming network traffic. Finally, the system raises an alert when it detects any intrusion.

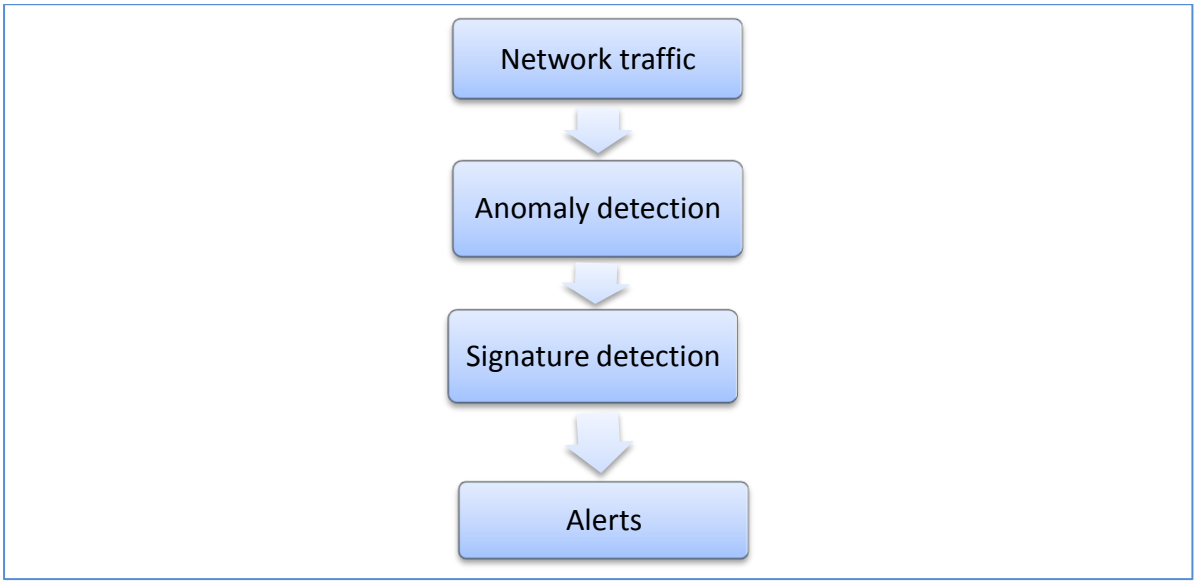


Figure 4.2: Schematic diagram of the system

For fulfilling the above objective, Snort which is signature based IDS and AnomalyDetection which is anomaly based IDS are taken. Figure 4.3 shows the methodology adopted to carry out the work. After installation of these systems, DARPA IDS Evaluation dataset is used as input to the system. Firstly, signature based detection is tested with the dataset and then the conjuncture of anomaly detection and signature detection is tested with the dataset. Alerts are generated which are stored in MySQL and a file names as alert. The output of both these approaches is compared based on various parameters.

Secondly, the DARPA IDS Evaluation dataset files are converted to CSV (Comma Separated Value) files. The labelling of data is done so as to test the data on WEKA. The labelling of data is done with the help of alert file. Lastly, the data is tested on WEKA for analysis the results are discussed.

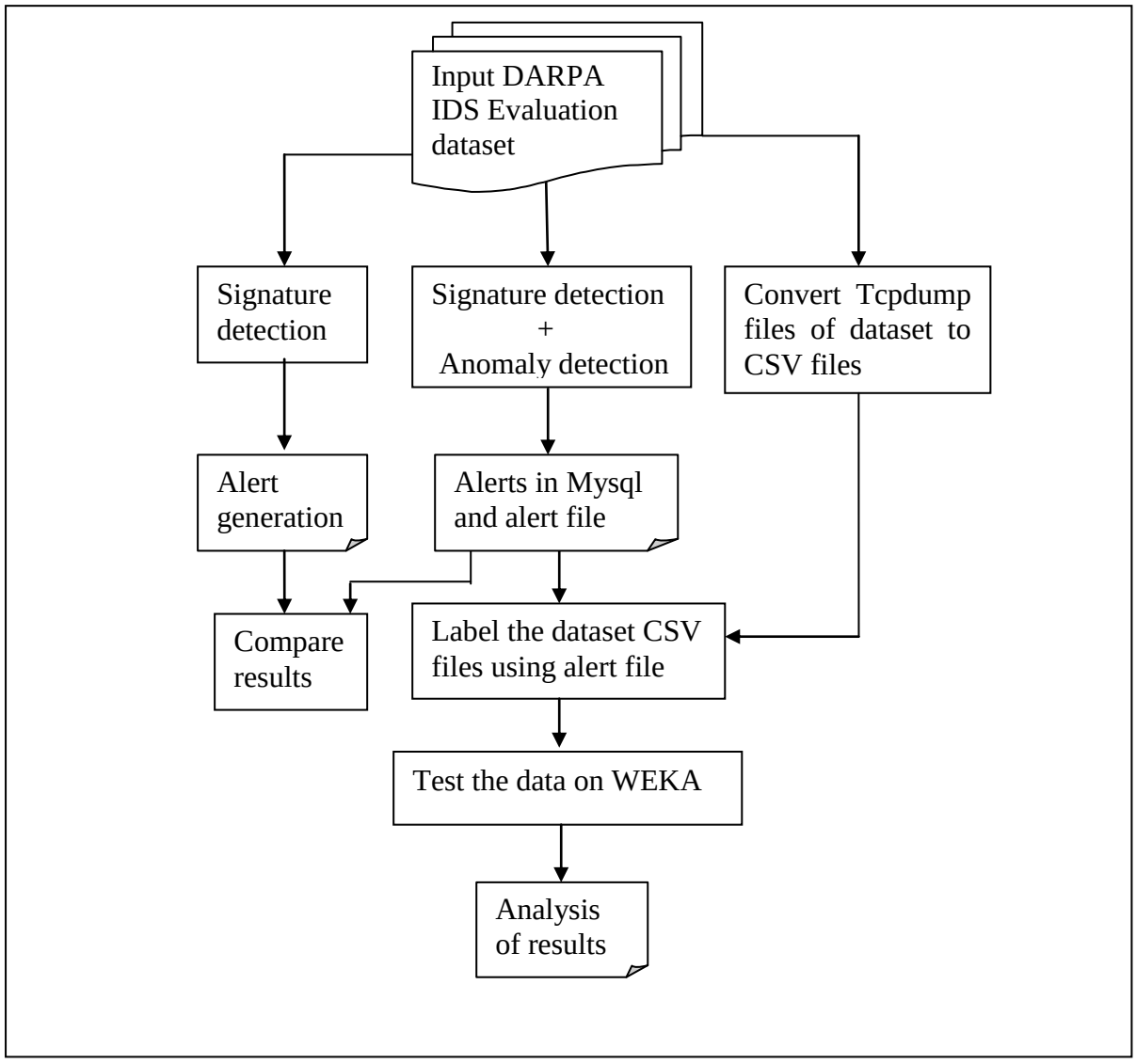


Figure 4.3: General design of workflow

4.3 Steps performed during configuration of Snort on Backtrack

Configuration of Snort on Backtrack involves the following steps:

Step 1: Snort is already configured on Backtrack .The snort.conf file defines how snort will run once the application is started.

The snort.conf file is edited as follows:

Step 1.1 vim /etc/snort/snort.conf

Step 1.2 Find the variable RULE_PATH and change to /etc/snort/rules

Step 1.3 Find output and comment out any output modules currently on.

Step 1.4 Find "output log_unified". Insert the following below it: output unified2: filename snort.log, limit=128

Step 2: MySQL will serve as the database for the snort application. Barnyard2 is downloaded from the url *http://www.secureixlive.com/download/barnyard2/barnyard2-1.9.tar.gz* and files are extracted.

Step 2.1 The first step is to turn on the MySQL service. Goto Backtrack-> services-> MySQLD-> start MySQL.

Step 2.2 On the command line interface type the following:

- mysql –u root –p
- Enter password:
- create database snort;
- exit
- mysql –D snort –u root -p </root/barnyard/barnyard/barnyard2-1.9/schemas/create_mysql

Step 2.3 Now check to see if the Snort database has been correctly installed by using the following commands.

- mysql –u root –p
- Enter password:
- use snort;
- show tables;

The Snapshot 4.1 shows the list of tables that will appear. Snort database consists of 16 tables.

```
mysql> use snort;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+-----------------+
| Tables_in_snort |
+-----------------+
| data            |
| detail          |
| encoding        |
| event           |
| icmphdr         |
| iphdr           |
| opt             |
| reference       |
| reference_system |
| schema          |
| sensor          |
| sig_class       |
| sig_reference   |
| signature       |
| tcphdr          |
| udphdr          |
+-----------------+
16 rows in set (0.00 sec)

mysql>
```

Snapshot 4.1: Tables in Snort database

Step 3: Barnyard2 was written to take over the various output processing tasks so that Snort could spend more resources on processing packets. The following steps are performed to configure Barnyard2.

Step 3.1 cd /usr/local

- tar zxvf /root/Desktop/barnyard2-1.9.tar.gz
- cd barnyard2-1.9
- ./configure --with-mysql && make && make install
- cp etc/barnyard2.conf /etc/snort

Step 3.2 Now open the barnyard.conf file to make changes as follows:

- vim /etc/snort/barnyard2.conf
- replace “thor” with “localhost”
- Look for config interface,make it eth0
- Edit the mysql line to read the following in output database section.

  output database: alert, mysql, user=snort password=password dbname=snort host=localhost

Step 3.3 In the Command line interface, type the following

snort -c /etc/snort/snort.conf

Step 3.4 Open a second Command line interface and type the following

- ls -la /var/log/snort. The 10 digit suffix on snort.log is copied. If there is more than one file, latest file is copied.
- vim /var/log/snort/barnyard.waldo

Step 3.5 Enter the following in barnyard.waldo file, then save and exit.

/var/log/snort
snort.log
1367945518
0

1367945518 is 10 digit suffix on snort.log.

Step 3.6 The last step is to start Barnyard2 and check if it is properly installed.

/usr/local/bin/barnyard2 -c /etc/snort/barnyard2.conf -G
/etc/snort/gen-msg.map -S /etc/snort/sid-msg.map -d /var/log/snort -f
snort.log -w /var/log/snort/barnyard.waldo

4.4 Steps performed during installation of AnomalyDetection and Snort on Ubuntu

Configuration of Snort and AnomalyDetection on Ubuntu involves the following steps [60].

Step 1: To establish a segregate network using virtualization. Oracle VM VirtualBox is used to establish a segregate network and Ubuntu 12.04 operating system is installed on it.

Step 2: Configuring Snort in Ubuntu 12.04 platform requires some packages. All the dependant packages are grabbed from Synaptic *i.e.* System > Administration >Synaptic Package Manager. Manager searches for the following packages and if not present were installed.

- apache2
- php5
- php5-mysql
- php5-gd
- libpcap0.8-dev
- libpcre3-dev
- g++
- bison
- flex
- libpcap-ruby
- make
- autoconf
- libtool
- mysql-server
- libmysqlclient-dev

Two additional packages DAQ and Libdnet are downloaded from the following url *http://www.snort.org/downloads* and *http://libdnet.googlecode.com/files/libdnet-1.12.tgz* respectively and installed using the following command on command line interface.

./configure && make && make install

Step 3: The AnomalyDetection project files are downloaded from the link *https://bitbucket.org/AnomalyDetection/preprocessor* and Snort is downloaded from the link *www.snort.org*. The downloaded files are extracted and kept in home directory. The AnomalyDetection project consists of preprocessor and profile generator files.

Step 4: Copy spp_anomalydetection.c and spp_anomalydetection.h from /home/preprocessor/src/preprocessors/ directory to /home/snort/src/preprocessors. This is done by typing the following on the command line interface.

cp /home/preprocessor/src/preprocessors/spp_anomalydetection.* /home/snort/src/preprocessors

Step 5: Open plugbase.c file located at filepath /home/snort/src/

This file has to be modified.

- In section /* built-in preprocessors */ , add the following header #include "preprocessors/spp_anomalydetection.h"
- In function void RegisterPreprocessors(void), add the following line SetupAnomalyDetection();

Step 6: The next step is to update Makefile.in file. The file is located at /home/snort-x.x.x/src/preprocessors/

- In the end of 'libspp_a_SOURCES' section add the following line spp_ anomalydetection.c spp_anomalydetection.h
- In the end of am_libspp_a_OBJECTS add spp_anomalydetection.$(OBJEXT)

Step 7: The definition of rules that are part of AnomalyDetection is contained in the file located at /home/preprocessor/src/generators.h. These definitions are added to file at location /home/snort/src/generators.h

Step 8: All the rules from /home/preprocessor/preproc_rules/preprocessor.rules file is added to preprocessor.rules file located at /home/snort/preproc_rules/preprocessor.rules [22].

Step 9: This step involves compiling the whole project. Go to snort directory and type the following command in command line interface under root privileges.

```
cd snort
./configure && make && make install
mkdir /var/log/snort
mkdir /var/snort
groupadd snort
useradd -g snort snort
chown snort:snort /var/log/snort
```

Step 10: The next step consists of downloading the latest rule set for Snort. It can be downloaded from the following url *https://www.snort.org/snort-rules*. snortrules-snapshot 2910.tar.gz file is downloaded and the following commands are issued on command line.

tar zxvf snortrules-snapshot-2910.tar.gz -C /usr/local/snort

mkdir /usr/local/snort/lib/snort_dynamicrules

cp /usr/local/snort/so_rules/precompiled/Ubuntu-10-4/i386/2.9.1.0/* /usr/local/snort/lib/snort_dynamicrules

touch /usr/local/snort/rules/white_list.rules

touch /usr/local/snort/rules/black_list.rules

ldconfig

Step 11: The next step is to make changes in the configuration file of Snort located at /usr/local/snort/etc/snort.conf

The lines given below are edited. These lines were changed to the following.

var WHITE_LIST_PATH /usr/local/snort/rules

var BLACK_LIST_PATH /usr/local/snort/rules

dynamicpreprocessor directory /usr/local/snort/lib/snort_dynamicpreprocessor/

dynamicengine /usr/local/snort/lib/snort_dynamicengine/libsf_engine.so

dynamicdetection directory /usr/local/snort/lib/snort_dynamicrules

The following lines are added in the output plugin section.

output unified2: filename snort.u2, limit 128

output alert_csv: /var/log/alert.csv default

Step 12: Downloading and installing Barnyard2

Barnyard2 improves the efficiency of Snort by reducing the load on detection engine. It reads Snort's unified logging output files and logs them into the database. Barnyard2 is downloaded from the url *http://www.secureixlive.com/download/barnyard2/barnyard2-1.9.tar.gz* .

Step 12.1: Installation is done under root privileges as follows.

tar zxvf barnyard2-1.9.tar.gz

cd barnyard2-1.9

./configure --with-mysql

make

make install

cp etc/barnyard2.conf /usr/local/snort/etc

mkdir /var/log/barnyard2

chmod 666 /var/log/barnyard2

touch /var/log/snort/barnyard2.waldo

chown snort.snort /var/log/snort/barnyard2.waldo

Step 12.2: This step includes creating MySQL database and database schema. Type the following on command line. Here the password to be entered is MySQL password given during installation of MySQL server.

echo "create database snort;" | mysql -u root -p

mysql -u root -p -D snort < ./schemas/create_mysql

Step 12.3: The following step creates a MySQL user for Snort. The password to be entered here is a new password to be given to Snort user.

echo "grant create, insert, select, delete, update on snort.* to snort@localhost identified by 'YOURPASSWORD'" | mysql -u root -p

Step 12.4: Barnyard2 configuration file is modified as follows:

vi /usr/local/snort/etc/barnyard2.conf

The following lines in barnyard.conf file that are changed are given below.

config reference_file: /usr/local/snort/etc/reference.config

config classification_file: /usr/local/snort/etc/classification.config

config gen_file: /usr/local/snort/etc/gen-msg.map

config sid_file: /usr/local/snort/etc/sid-msg.map

config hostname: localhost

config interface: eth1

output database: log, mysql, user=snort password=YOURPASSWORD dbname=snort host=localhost

Instead of YOURPASSWORD use MySQL password given during installation of MySQL [60].

4.5 Experiments and results

The section covers the experiments performed during entire thesis. Experiments are broadly divided into three parts. The first part consists of the experiments is done on signature detection. The second part consists of experiments performed on conjuncture of anomaly and

signature detection. The experiments for analyzing the data on machine learning tool are performed in the third part of the implementation. The dataset for performing the above experiments is downloaded from the following url *http://www.ll.mit.edu/mission/communications/cyber/CSTcorpora/ideval/data/1999data.html* The inside sniffer traffic (inside.tcpdump.gz) week 4 having 4 files, and week 5 having 5 files is downloaded from the link mentioned above. These are uncompressed with gzip and renamed as follows.

Week 4 (test) in41 in43 in44 in45

Week 5 (test) in51 in52 in53 in54 in55

4.5.1 Experimentation on signature detection

Snort is run in network intrusion detection system (NIDS) mode. The week 4 and week 5 tcpdump files of inside sniffer traffic are given as input. Before running Snort, command to start Barnyard2 is given on another command line interface as follows.

/usr/local/bin/barnyard2 -c /etc/snort/barnyard2.conf -G /etc/snort/gen-msg.map -S /etc/snort/sid-msg.map -d /var/log/snort -f snort.log -w /var/log/snort/barnyard.waldo

The following command is typed on the command line interface to run Snort in NIDS mode and is shown in snapshot 4.2.

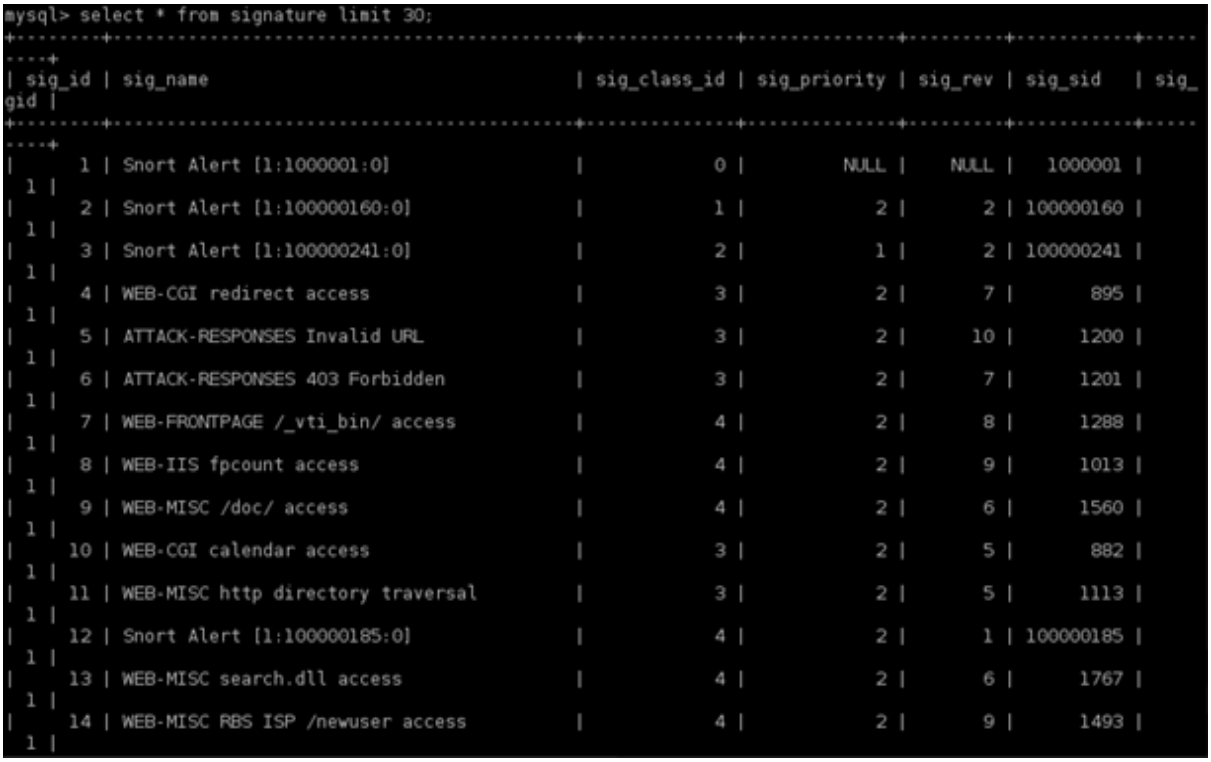
snort -r /root/inside_sniffer/filename.tcpdump -c /etc/snort/snort.conf

```
root@bt:~# snort -r /root/inside_sniffer/in55.tcpdump -c /etc/snort/snor
t.conf
Running in IDS mode

        --== Initializing Snort ==--
Initializing Output Plugins!
Initializing Preprocessors!
Initializing Plug-ins!
Parsing Rules file "/etc/snort/snort.conf"
PortVar 'HTTP_PORTS' defined :  [ 80 ]
PortVar 'SHELLCODE_PORTS' defined :  [ 0:79 81:65535 ]
PortVar 'ORACLE_PORTS' defined :  [ 1521 ]
PortVar 'FTP_PORTS' defined :  [ 21 ]
Tagged Packet Limit: 256
Loading dynamic engine /usr/lib/snort_dynamicengine/libsf_engine.so... d
one
Loading all dynamic preprocessor libs from /usr/lib/snort_dynamicpreproc
essor/...
  Loading dynamic preprocessor library /usr/lib/snort_dynamicpreprocesso
r//libsf_dns_preproc.so... done
  Loading dynamic preprocessor library /usr/lib/snort_dynamicpreprocesso
r//libsf_smtp_preproc.so... done
  Loading dynamic preprocessor library /usr/lib/snort_dynamicpreprocesso
r//libsf_ssh_preproc.so... done
  Loading dynamic preprocessor library /usr/lib/snort_dynamicpreprocesso
r//lib_sfdynamic_preprocessor_example.so... done
  Loading dynamic preprocessor library /usr/lib/snort_dynamicpreprocesso
r//libsf_ftptelnet_preproc.so... done
  Loading dynamic preprocessor library /usr/lib/snort_dynamicpreprocesso
r//libsf_ssl_preproc.so... done
  Loading dynamic preprocessor library /usr/lib/snort_dynamicpreprocesso
r//libsf_dce2_preproc.so... done
  Loading dynamic preprocessor library /usr/lib/snort_dynamicpreprocesso
r//libsf_dcerpc_preproc.so... done
  Finished Loading all dynamic preprocessor libs from /usr/lib/snort_dyn
```

Snapshot 4.2: Running Snort in NIDS mode

After running the Snort in NIDS mode, Snort stores the alerts/signatures generated by it in the Snort database. Snapshot 4.3 shows the Signature table of Snort database. The rest of the files of dataset are tested in the similar fashion.



```
mysql> select * from signature limit 30;
+--------+-----------------------------------------+-------------+--------------+---------+-----------+-----
----+
| sig_id | sig_name                                | sig_class_id | sig_priority | sig_rev | sig_sid   | sig_
gid |
+--------+-----------------------------------------+-------------+--------------+---------+-----------+-----
----+
|      1 | Snort Alert [1:1000001:0]               |            0 |         NULL |    NULL |   1000001 |
  1 |
|      2 | Snort Alert [1:100000160:0]             |            1 |            2 |       2 | 100000160 |
  1 |
|      3 | Snort Alert [1:100000241:0]             |            2 |            1 |       2 | 100000241 |
  1 |
|      4 | WEB-CGI redirect access                 |            3 |            2 |       7 |       895 |
  1 |
|      5 | ATTACK-RESPONSES Invalid URL            |            3 |            2 |      10 |      1200 |
  1 |
|      6 | ATTACK-RESPONSES 403 Forbidden          |            3 |            2 |       7 |      1201 |
  1 |
|      7 | WEB-FRONTPAGE /_vti_bin/ access         |            4 |            2 |       8 |      1288 |
  1 |
|      8 | WEB-IIS fpcount access                  |            4 |            2 |       9 |      1013 |
  1 |
|      9 | WEB-MISC /doc/ access                   |            4 |            2 |       6 |      1560 |
  1 |
|     10 | WEB-CGI calendar access                 |            3 |            2 |       5 |       882 |
  1 |
|     11 | WEB-MISC http directory traversal        |            3 |            2 |       5 |      1113 |
  1 |
|     12 | Snort Alert [1:100000185:0]             |            4 |            2 |       1 | 100000185 |
  1 |
|     13 | WEB-MISC search.dll access              |            4 |            2 |       6 |      1767 |
  1 |
|     14 | WEB-MISC RBS ISP /newuser access        |            4 |            2 |       9 |      1493 |
  1 |
```

Snapshot 4.3: Signatures stored into MySQL

4.5.2 Experimentation on anomaly + signature detection

- Firstly a network based log file has to be created. The log file will contain a profile of network traffics characteristics. Create a log file named ADLog60.txt at location /var/log/snort. Edit the snort.conf file to add the following line at the end of the preprocessor's section.

  `preprocessor AnomalyDetection: LogPath /var/log/snort log time 60`

- The next step is to run Snort and generate log data. Snort is run in NIDS mode. This would create a log with a number of entries saved in /var/log/snort directory.

  `sudo snort –c /usr/local/snort/etc/snort.conf -i eth0`

The Snapshot 4.4 shows the output as a result of running Snort in NIDS mode. The messages shown in the Snapshot appear on the screen. The data is being logged into the log file. The log file will now contain number of entries located at /var/log/snort.

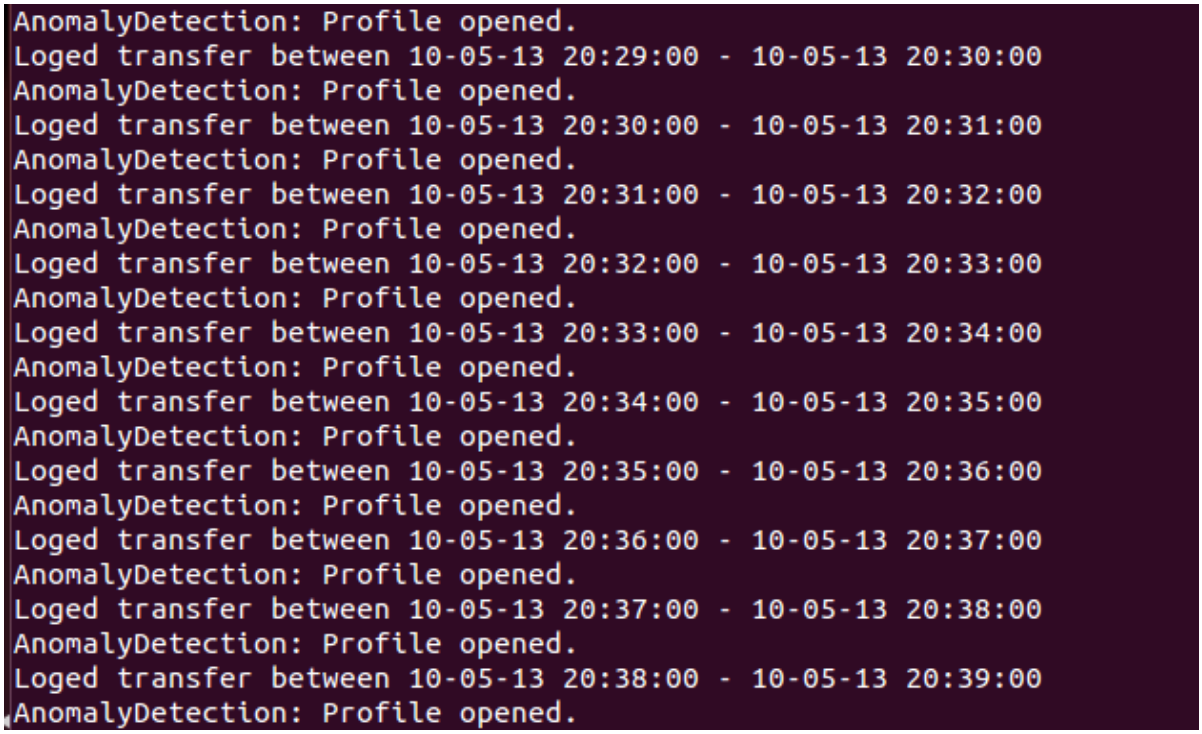


Figure 4.4: Entries logged into log file

- The next step is to run the profile generation script. This command creates the profile.txt file which is a CSV (comma separated file). This CSV file will be used by snort.conf file. The following commands are issued on CLI.

  cd /home/ADprofilegenerator

  ./ad_profilegenerator.r -m AVG --avg 'WEEKLY,1' -l Log_Data.txt -p profile.txt -e evaluator.txt -P pattern.txt

- The last step is to test the system with DARPA IDS Evaluation dataset. Before running snort in NIDS mode, edit snort.conf file and add the following line in the file as shown in the Snapshot 4.5.

  preprocessor AnomalyDetection: ProfilePath /etc/profile.txt LogPath /var/log/snort alert log time 60

```
# POP preprocessor. For more information see README.pop
preprocessor pop: \
   ports { 110 } \
   b64_decode_depth 0 \
   qp_decode_depth 0 \
   bitenc_decode_depth 0 \
   uu_decode_depth 0

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

#anomalydetection preprocessor
preprocessor AnomalyDetection: ProfilePath /etc/profile.txt alert log time 60
##################################################
```

Figure 4.5: Adding AnomalyDetection as preprocessor in snort.conf file

The line added is explained below.

ProfilePath: The path to profile file is put after this keyboard. Default is /etc/profile.txt.

LogPath: The full path to directory which contains logs files is mentioned after this keyword. Default is /var/log/snort.

Alert: if alert is set, preprocessor will be able to report anomalies.

Log: if log is set, preprocessor will be able to log network traffic to file.

Time: It indicates the time interval after which the traffic is logged. Default value is 600 seconds [22].

Now the system is ready to run in NIDS mode. The Preprocessor module will read the profile file generated above. For the purpose of logging alerts in MySQL, start Barnyard2 using the following command as shown in Snapshot 4.6.

```
root@tj-VirtualBox:/# /usr/local/bin/barnyard2 -c /usr/local/snort-2.9.1/etc/barn
yard2.conf -G /usr/local/snort-2.9.1/etc/gen-msg.map -S /usr/local/snort-2.9.1/et
c/sid-msg.map -d /var/log/snort -f snort.u2 -w /var/log/snort/barnyard2.waldo
Running in Continuous mode

        --== Initializing Barnyard2 ==--
Initializing Input Plugins!
Initializing Output Plugins!
Parsing config file "/usr/local/snort-2.9.1/etc/barnyard2.conf"
```

Snapshot 4.6: Start Barnyard2

Snort is run in NIDS mode on another command line interface using following command.

Snort -r /home/data/in41.tcpdump -c /usr/local/snort/etc/snort.conf

Similarly all other tcpdump files are read, processed and the alerts are logged in MySQL. The Snapshot 4.7 shows the AnomalyDetection statistics shown by Snort after processing all the packets in the file.

```
===============================================================================
AnomalyDetection statistics:
          Overall packets: 3738928
     Other than IP packets: 9223
     Number of TCP packets: 3046897
    Number of IP datagrams: 1386
   Number of UDP datagrams: 674102
    Number of ICMP packets: 1249
     Number of ARP packets: 6071
     Number of ARP request: 3057
       Number of ARP reply: 3014
                  ARP diff: 43

        Traffic in LAN TCP: 3046897,                                    UDP:
674102,                                    ICMP: 0
   Traffic loged in /var/log/snort/ADLog60.txt
===============================================================================
```

Snapshot 4.7: Running Snort with AnomalyDetection in NIDS mode

4.5.3 Results of experimentation

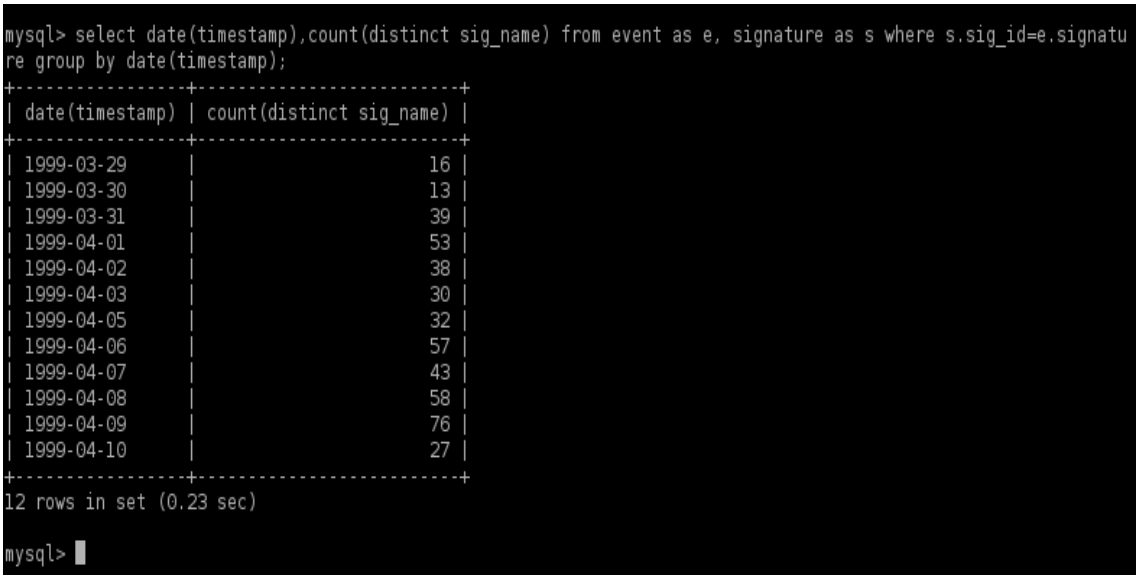
The alerts generated due evaluation of Snort and Snort with AnomalyDetection on the DARPA IDS Evaluation dataset are stored in the Snort database of MySQL. The signatures stored are retrieved from the tables of Snort database using MySQL queries. For the purpose of comparing the performance of the signature detection with signature + anomaly detection, the parameters used are given below. Same MySQL query is executed on both the approaches to compare the results.

i) Number of distinct alerts generated date wise

The output of the above query can be generated by joining the event and the signature table. The output of the query is shown in Snapshot 4.8 and Snapshot 4.9. The output shows the date and the no. of alerts generated on that date that are distinct. The results

shown are fourth and fifth week of DARPA IDS Evaluation dataset. The following MySQL query is executed to obtain the no. of distinct alerts generated date wise.

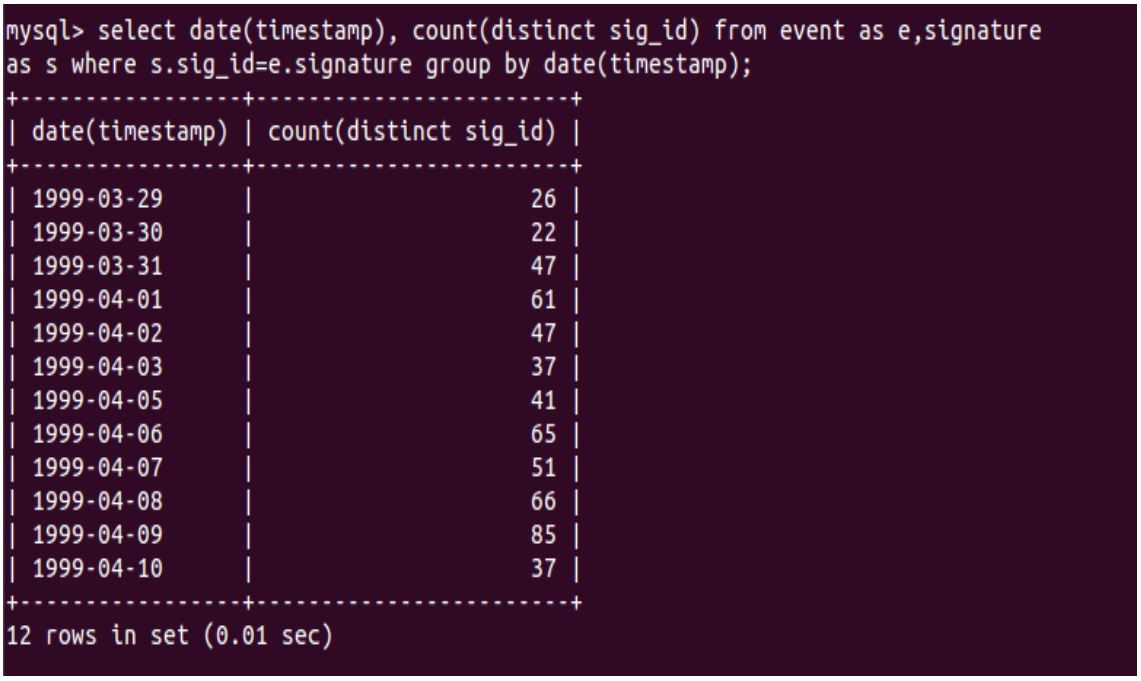
```
select date(timestamp), count(distinct sig_name) from event as e, signature as s where s.sig_name=e.signature group by date(timestamp);
```



```
mysql> select date(timestamp),count(distinct sig_name) from event as e, signature as s where s.sig_id=e.signatu
re group by date(timestamp);
+----------------+------------------------+
| date(timestamp) | count(distinct sig_name) |
+----------------+------------------------+
| 1999-03-29     |                     16 |
| 1999-03-30     |                     13 |
| 1999-03-31     |                     39 |
| 1999-04-01     |                     53 |
| 1999-04-02     |                     38 |
| 1999-04-03     |                     30 |
| 1999-04-05     |                     32 |
| 1999-04-06     |                     57 |
| 1999-04-07     |                     43 |
| 1999-04-08     |                     58 |
| 1999-04-09     |                     76 |
| 1999-04-10     |                     27 |
+----------------+------------------------+
12 rows in set (0.23 sec)

mysql>
```

Snapshot 4.8: No. of distinct alerts listed date wise (signature detection)



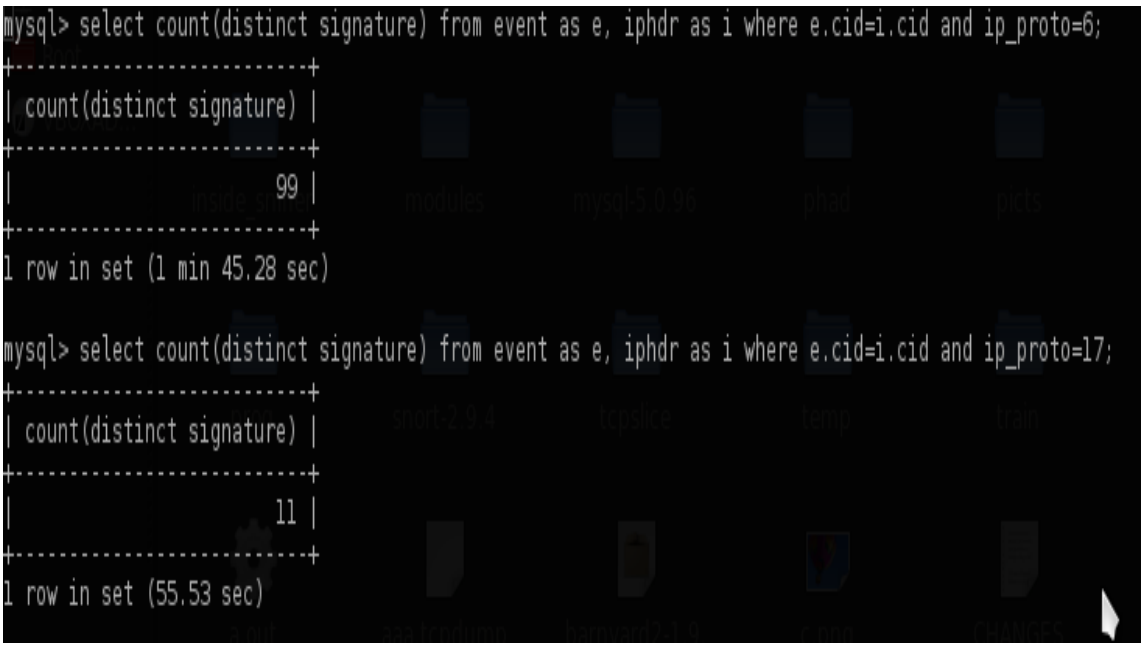
```
mysql> select date(timestamp), count(distinct sig_id) from event as e,signature
as s where s.sig_id=e.signature group by date(timestamp);
+----------------+----------------------+
| date(timestamp) | count(distinct sig_id) |
+----------------+----------------------+
| 1999-03-29     |                   26 |
| 1999-03-30     |                   22 |
| 1999-03-31     |                   47 |
| 1999-04-01     |                   61 |
| 1999-04-02     |                   47 |
| 1999-04-03     |                   37 |
| 1999-04-05     |                   41 |
| 1999-04-06     |                   65 |
| 1999-04-07     |                   51 |
| 1999-04-08     |                   66 |
| 1999-04-09     |                   85 |
| 1999-04-10     |                   37 |
+----------------+----------------------+
12 rows in set (0.01 sec)
```

Snapshot 4.9: No. of distinct alerts listed date wise (anomaly + signature detection)

ii) Number of alerts generated protocol wise

Another parameter for comparison is signatures generated protocol wise. The data is extracted from the iphdr and event table by applying join on these tables. The no. of signatures generated by the packets containing TCP (Transmission Control Protocol) and UDP (User Datagram Protocol) is obtained the following query:

```
select count(distinct signature) from event as e, iphdr as i where e.cid=i.cid and ip_proto=6;
```



Snapshot 4.10: Alerts listed protocol wise (signature detection)

```
mysql> select count(distinct signature) from event as e, iphdr as i where e.cid=
i.cid and ip_proto=6;
+---------------------------+
| count(distinct signature) |
+---------------------------+
|                       111 |
+---------------------------+
1 row in set (3 hours 28 min 11.56 sec)

mysql> select count(distinct signature) from event as e, iphdr as i where e.cid=
i.cid and ip_proto=17;
+---------------------------+
| count(distinct signature) |
+---------------------------+
|                        15 |
+---------------------------+
1 row in set (8 min 19.46 sec)
```

Snapshot 4.11: Alerts listed protocol wise (anomaly + signature detection)

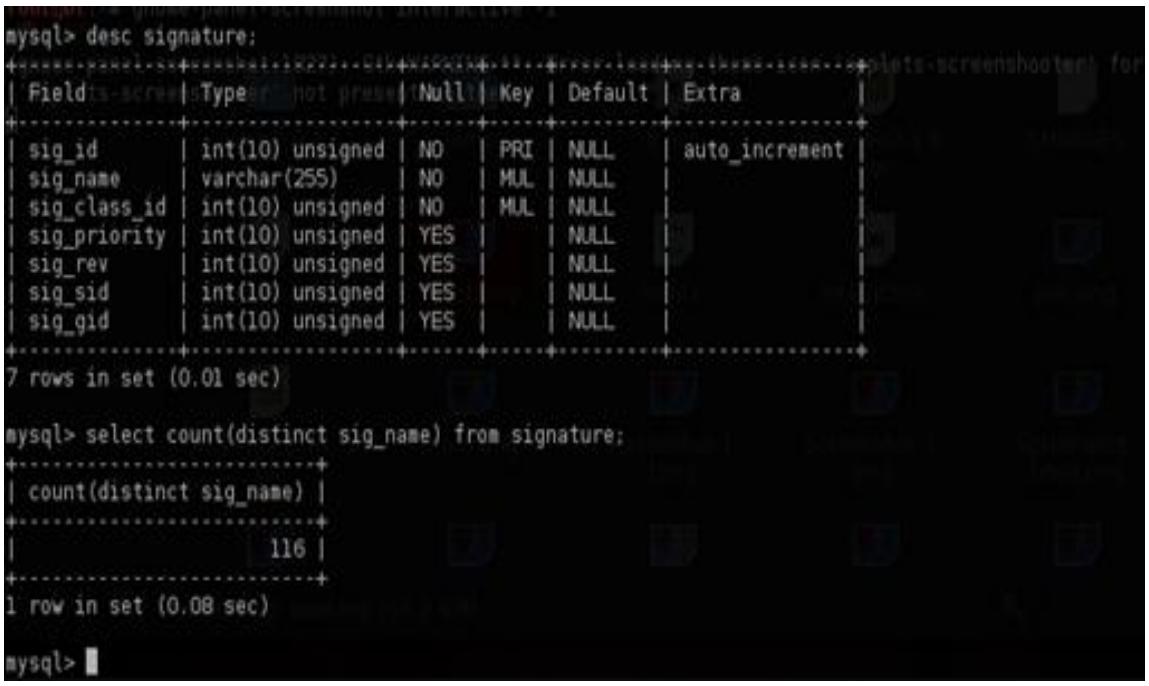
The Snapshot 4.10 and Snapshot 4.11 show the output of the query. In the ip_proto field, 6 represents TCP and 17 represents UDP.

iii) Total number of attacks detected

This is the most important parameter which describes the detection rate of the system. The signatures generated by the system means the no. of attacks detected by IDS. It depicts the real performance of an IDS. All the signatures generated are stored in the signature table which is obtained by the following query.

```
Select count (distinct sig_name) from signature;
```

The output of the above query is shown in Snapshot 4.12 and Snapshot 4.13. Signature detection detects 116 attacks whereas anomaly + signature detection detects 123 attacks which is more than signature detection.



Snapshot 4.12: No. of attacks detected (signature detection)

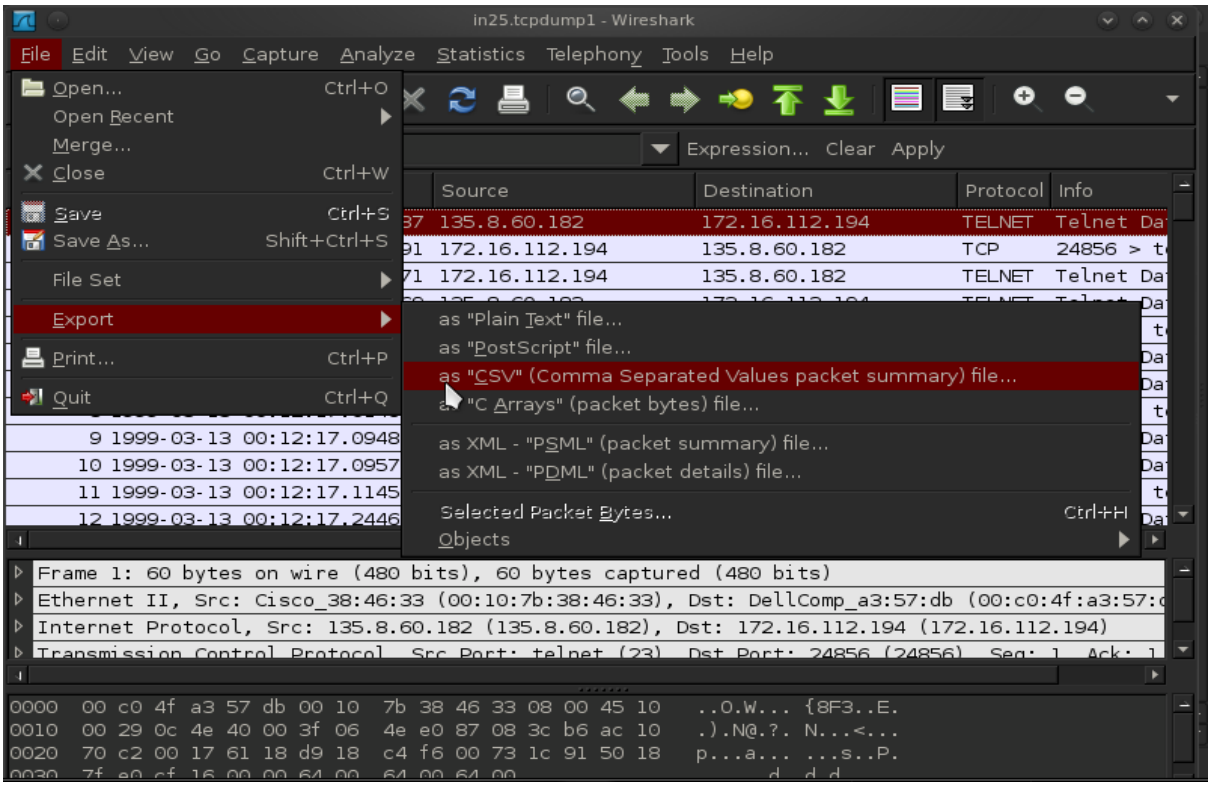
Snapshot 4.13: No. of attacks detected (anomaly + signature detection)

4.5.4 Data preparation

The output obtained as a result of the experiments performed on the IDS discussed above is to be tested on machine learning tool. For this purpose, the data has to be changed into the required format. This section explains the steps to prepare the data to be given as input to the machine learning tool, WEKA.

i) Converting tcpdump data into CSV files

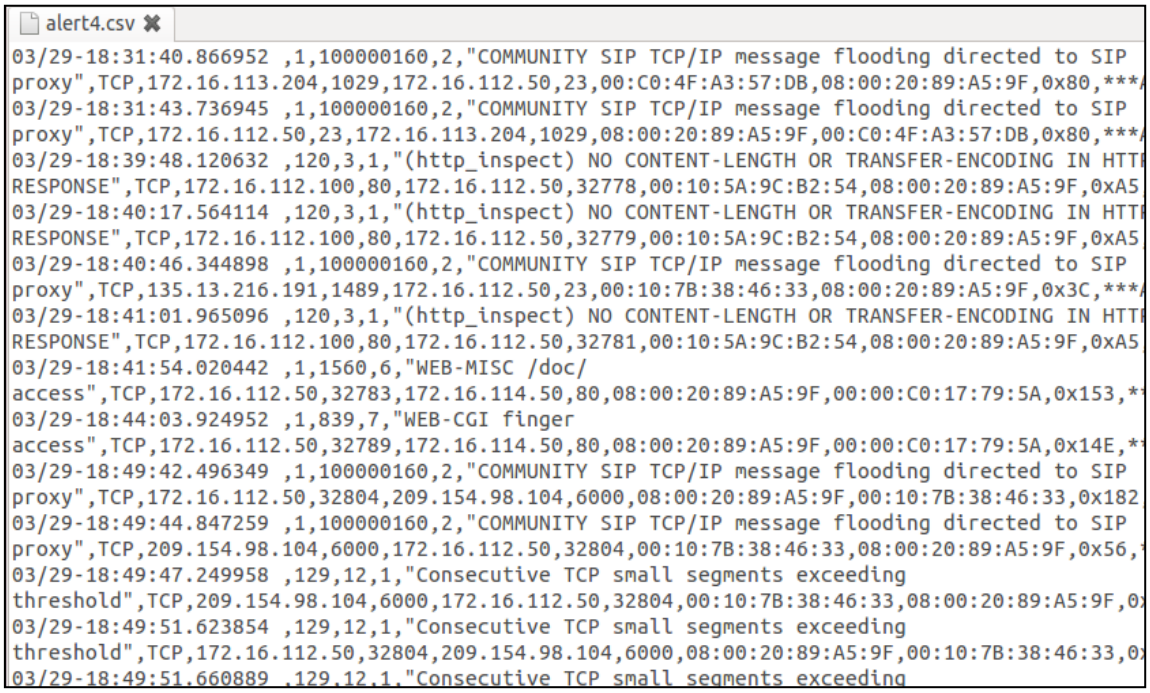
The tcpdump files of DARPA dataset are converted to CSV files using Wireshark. The tcpdump files are opened in Wireshark and then converted to CSV files. The Snapshot 4.14 shows the conversion of tcpdump file into CSV file in Wireshark. The converted CSV files contain network packets from the DARPA IDS Evaluation dataset. It contains a one-line summary for each packet captured, including the packet number assigned by Wireshark, the time at which the packet was captured, the packet's source and destination addresses, the protocol type, and protocol-specific information contained in the packet. The protocol type field lists the highest level protocol that sent or received this packet. After opening the tcpdump file in Wireshark, convert it to CSV file by clicking on File → Export →as "CSV" file.



Snapshot 4.14: Conversion of tcpdump file to CSV file

ii) Labelling the data

In order to label the packets of DARPA IDS Evaluation dataset, the signatures generated by the anomaly + signature based IDS is used. The alerts were logged into the alert file which is a comma separated value (CSV) file after running the anomaly + signature based IDS on DARPA IDS Evaluation dataset. The Snapshot 4.15 shows the alert.csv file containing the logged alerts. The file contains the timestamp of the packet, signature id of the rule that was triggered, msg rule option tells the logging and alerting engine the message to print along with a packet dump or to an alert, IP address of source, IP address of destination, protocol type, source port, destination port and various values from the fields of the IP header. This file is used to label the CSV files of the dataset. Thus if a packet did not trigger any attack alerts, it is labelled as *Normal*. If a packet triggered an attack alert, it is labelled as *Attack.*



Snapshot 4.15: Alert.csv file containing alerts

The procedure of labelling the data is explained below.

The work is accomplished by importing the data files and alert file into MS Access. The fields present in the data files are no., Time, source, destination, protocol, info. The similar fields in the alert file are timestamp, source, destination and protocol. The packets in the data file were labelled by comparing the timestamp, source, destination, protocol from both the

alert file and the data file. The field “class” is created in the table and the values in this field are inserted by using the following update query.

```
UPDATE alert, in41 SET in41.class = "attack" WHERE in41.time=alert.timestamp and in41.source=alert.source and in41.destination=alert.destination and in41.protocol=alert.protocol;
```

The rest of the data is labelled as “normal”. Now the data has been categorized as “normal” and “attack”. The result of the above MySQL query is shown in Snapshot 4.16. The above mentioned procedure is one way of labelling the data. There is another way of labelling the data. The first word of each alert message is used as the label names, this essentially corresponds to the name of rule base used for the generation of attacks (e.g. http_inspect, ftp_telnet, WEB-CGI). The data labelled using the first word of the alert message is shown in Snapshot 4.17.

No	Time	Source	Destination	Protocol	Info	class
675	1999-03-29 18::	172.16.112.50	172.16.113.204	TELNET	Telnet Data ...	normal
676	1999-03-29 18::	172.16.113.204	172.16.112.50	TCP	solid-mux > te	normal
677	1999-03-29 18::	172.16.113.204	172.16.112.50	TELNET	Telnet Data ...	normal
678	1999-03-29 18::	172.16.112.50	172.16.113.204	TELNET	Telnet Data ...	normal
679	1999-03-29 18::	172.16.113.204	172.16.112.50	TCP	solid-mux > te	attack
680	1999-03-29 18::	172.16.113.204	172.16.112.50	TELNET	Telnet Data ...	normal
681	1999-03-29 18::	172.16.112.50	172.16.113.204	TELNET	Telnet Data ...	normal
682	1999-03-29 18::	172.16.113.204	172.16.112.50	TCP	solid-mux > te	attack
683	1999-03-29 18::	172.16.113.204	172.16.112.50	TELNET	Telnet Data ...	normal
684	1999-03-29 18::	172.16.112.50	172.16.113.204	TELNET	Telnet Data ...	normal
685	1999-03-29 18::	172.16.113.204	172.16.112.50	TCP	solid-mux > te	attack
686	1999-03-29 18::	172.16.113.204	172.16.112.50	TELNET	Telnet Data ...	normal
687	1999-03-29 18::	172.16.112.50	172.16.113.204	TELNET	Telnet Data ...	normal
688	1999-03-29 18::	172.16.113.204	172.16.112.50	TCP	solid-mux > te	attack
689	1999-03-29 18::	172.16.113.204	172.16.112.50	TELNET	Telnet Data ...	normal
690	1999-03-29 18::	172.16.112.50	172.16.113.204	TELNET	Telnet Data ...	normal
691	1999-03-29 18::	172.16.113.204	172.16.112.50	TCP	solid-mux > te	attack
692	1999-03-29 18::	172.16.113.204	172.16.112.50	TELNET	Telnet Data ...	normal
693	1999-03-29 18::	172.16.112.50	172.16.113.204	TELNET	Telnet Data ...	normal
694	1999-03-29 18::	172.16.113.204	172.16.112.50	TCP	solid-mux > te	attack
695	1999-03-29 18::	172.16.113.204	172.16.112.50	TELNET	Telnet Data ...	normal
696	1999-03-29 18::	172.16.112.50	172.16.113.204	TELNET	Telnet Data ...	normal

Snapshot 4.16: Labelling of data in MS Access (1)

No	Time	Source	Destination	Protocol	Info	class
6546	1999-03-29 18:4	135.13.216.191	172.16.112.50	TELNET	Telnet Data ...	normal
6547	1999-03-29 18:4	172.16.112.50	135.13.216.191	TELNET	Telnet Data ...	normal
6548	1999-03-29 18:4	172.16.112.50	172.16.114.50	TCP	32783 > http [S	WEB-MISC
6549	1999-03-29 18:4	172.16.114.50	172.16.112.50	TCP	http > 32783 [S	normal
6550	1999-03-29 18:4	172.16.112.50	172.16.114.50	TCP	32783 > http [A	WEB-MISC
6551	1999-03-29 18:4	172.16.112.50	172.16.114.50	HTTP	GET /doc/expl	normal
6552	1999-03-29 18:4	172.16.114.50	172.16.112.20	DNS	Standard quer	normal
6553	1999-03-29 18:4	172.16.112.20	172.16.114.50	DNS	Standard quer	normal
6554	1999-03-29 18:4	135.13.216.191	172.16.112.50	TCP	dmdocbroker >	normal
6555	1999-03-29 18:4	172.16.112.50	135.13.216.191	TELNET	Telnet Data ...	normal
6556	1999-03-29 18:4	172.16.114.50	172.16.112.50	TCP	http > 32783 [A	normal
6557	1999-03-29 18:4	135.13.216.191	172.16.112.50	TCP	dmdocbroker >	normal
6558	1999-03-29 18:4	172.16.114.50	172.16.112.50	HTTP	HTTP/1.0 200 C	normal
6559	1999-03-29 18:4	172.16.114.50	172.16.112.50	TCP	http > 32783 [F	normal
6560	1999-03-29 18:4	172.16.112.50	172.16.114.50	TCP	32783 > http [A	WEB-MISC
6561	1999-03-29 18:4	172.16.112.50	135.13.216.191	TELNET	Telnet Data ...	normal
6562	1999-03-29 18:4	172.16.112.50	172.16.114.50	TCP	32783 > http [F	WEB-MISC
6563	1999-03-29 18:4	172.16.114.50	172.16.112.50	TCP	http > 32783 [A	normal
6564	1999-03-29 18:4	135.13.216.191	172.16.112.50	TCP	dmdocbroker >	normal
6565	1999-03-29 18:4	172.16.112.50	135.13.216.191	TELNET	Telnet Data ...	normal

Snapshot 4.17: Labelling of data in MS Access (2)

The dataset prepared contains millions of records. It is not possible to test the whole data on WEKA. A small set of data is used to evaluate the results in WEKA because the whole data is too large to be processed by WEKA. This occurs due to memory limitations. It becomes quite computationally expensive and the complexity will increase due to large dataset. So only a subset of the dataset is taken for testing. The subset consists of the data from 18th hour on 29th April. It consists of 23995 packets. Now there are two datasets *i.e.* dataset (1) and dataset (2) having the identical records but having only different labels. The details of the obtained labels is shown in Table 4.1 and Table 4.2.

Table 4.1: Class distribution in the dataset (1)

Label	**Count**	**Percentage**
Normal	23731	98.89
Attack	264	1.1

Table 4.2: Class distribution in the dataset (2)

Label	**Count**	**Percentage**
Normal	23731	98.89
Community	98	0.004
Http inspect	6	0.00025
WEB-MISC	4	0.00016
WEB-CGI	5	0.0002
TCP small segment	147	0.006
Ftp telnet	4	0.00016

4.5.5 Experiments performed on WEKA

After the data is categorized, the data is analyzed on a machine learning tool known as WEKA. WEKA is chosen as it is a very robust open source machine learning workbench which has more than 80 classifier algorithms to choose from. It is quite a challenging task to choose the right algorithm. Classification algorithms or classifiers give results by identifying patterns in an existing dataset. The algorithms take pre-classified data as input. They learn the patterns in the data and create simple rules to differentiate between the various types of data in the pre-classified data set. The figure 4.4 shows the classifier inputs and outputs.

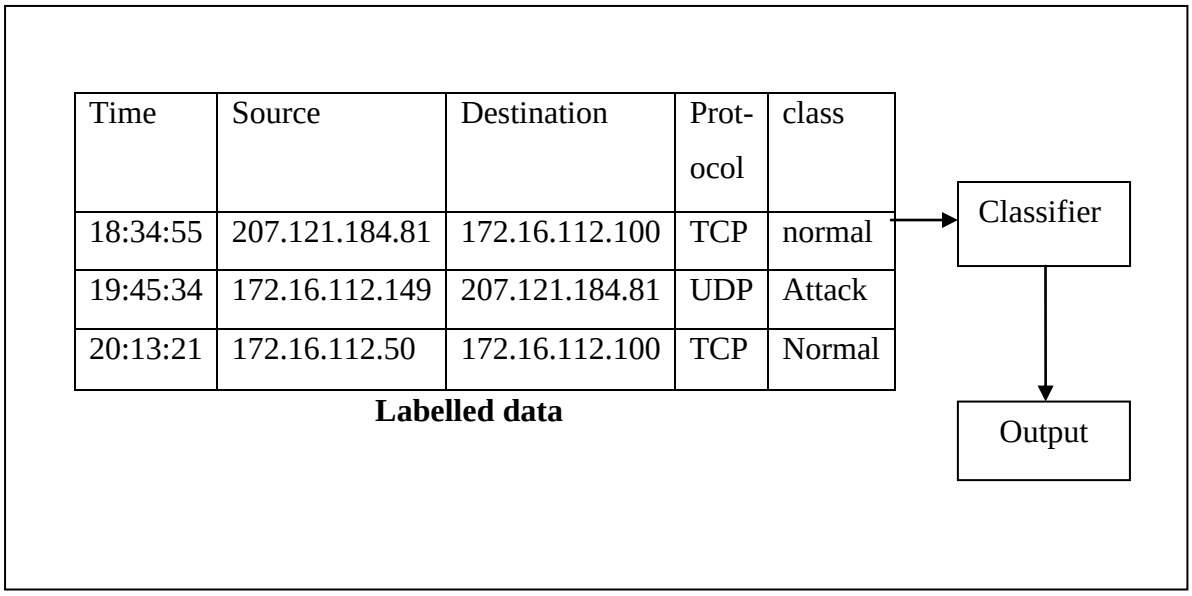
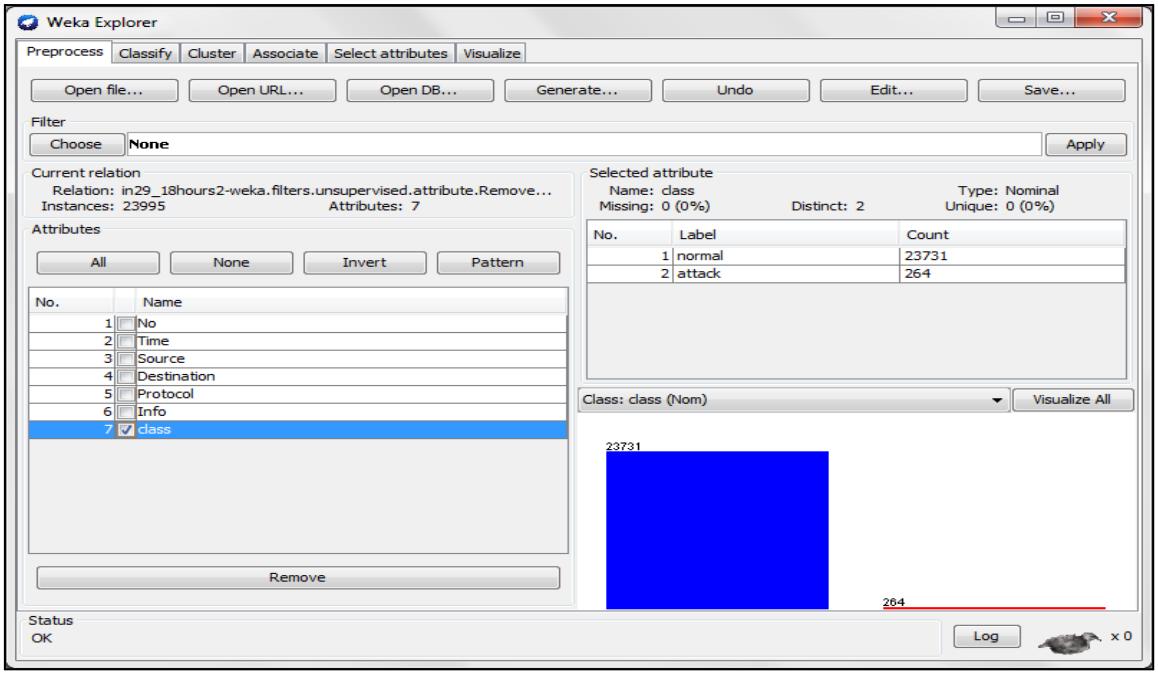


Figure 4.4: Data mining process

The following are the steps that are performed to analyze the data in WEKA.

i) Preprocessing

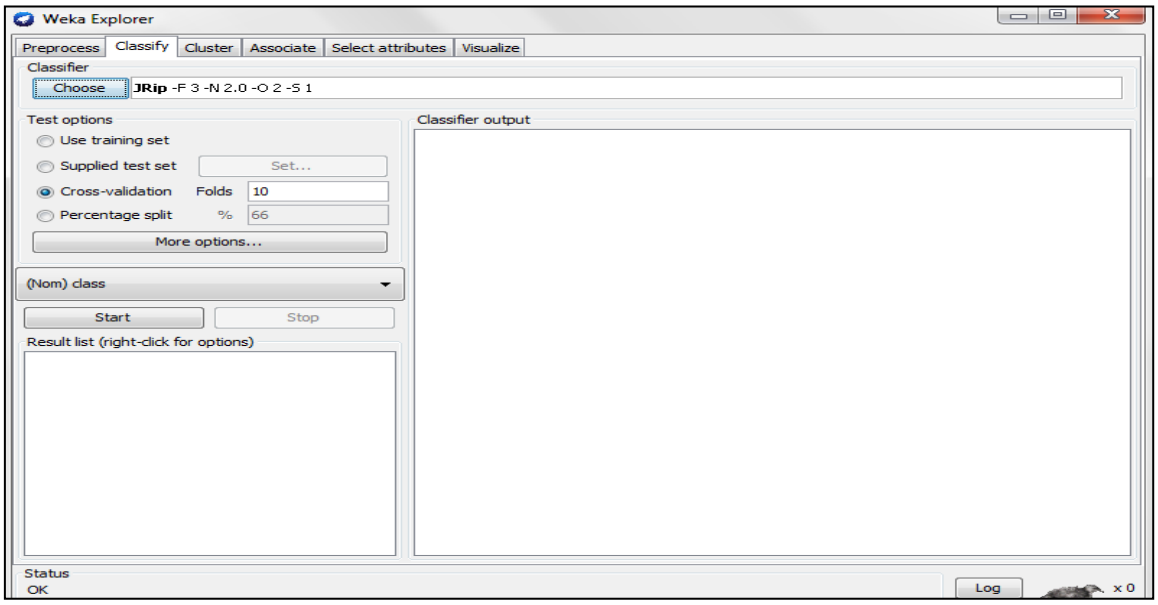
Preprocessing means to choose and modify the data being acted on. The labelled data is opened in WEKA and the attributes are listed as shown in Snapshot 4.18. The attributes that are used for analysis purpose are Time, Source, Destination, Protocol and Info.



Snapshot 4.18: Preprocessing data in WEKA

ii) Classification

In the classify section, select the JRip classifier in the rules category. JRip classifier is used to test the dataset on WEKA. There are various test options in WEKA. Cross validation test option is used as test option. In Cross-validation, data is split into *k* subsets of equal size. Each subset in turn is used for testing and the remainder for training. This is called *k-fold cross-validation.* Snapshot 4.19 shows using the 10 fold cross validation test option and JRip classifier on prepared dataset.



Snapshot 4.19: Applying Classifier on data

iii) Output

Snapshot 4.19 shows the run information of the classifier on the data. The analysis of the run information is done in Chapter 6.

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       23972
99.9041 %
Incorrectly Classified Instances        23
0.0959 %
Kappa statistic                          0.9562
Mean absolute error                      0.0012
Root mean squared error                  0.0292
Relative absolute error                  5.4986 %
Root relative squared error             27.971  %
Total Number of Instances            23995

=== Detailed Accuracy By Class ===

TP Rate FP Rate Precision Recall  F-Measure  ROC Area  Class
0.999   0.038     1        0.999     1          0.984    normal
0.962   0.001     0.951    0.962     0.957      0.984    attack

=== Confusion Matrix ===

     a     b   <-- classified as
 23718    13 |     a = normal
    10   254 |     b = attack
```

Snapshot 4.20: Run Information of the Classifier

4.6 Summary of the chapter

The chapter discusses the experimental setup, installation of Snort and AnomalyDetection for performing the experiments. The methodology used to carry out the experiments is presented. The experiments performed to implement the methodology are covered in detail and the results are shown in Snapshots.

Chapter 5

Results and Discussion

This chapter discusses the results of the experimentation illustrated in the previous chapter. Firstly, the results of signature detection are compared with anomaly + signature detection. The performance is evaluated on based on various parameters which are discussed in the next section. Secondly, analysis of the data on WEKA is done. The performance metrics on testing the data on WEKA include accuracy, true positive rate and false positive rate. The results are analyzed and discussed in the chapter.

5.1 Comparative analysis of signature detection with anomaly detection + signature detection

The approaches whose performance is to be discussed is given below

- Signature detection
- Anomaly + signature detection

The comparison of these two approaches can be performed on various scenarios. The following parameters are chosen and their performance is evaluated.

- Number of distinct alerts generated date wise
- Number of alerts generated protocol wise
- No. of attacks detected or detection rate

i) Number of distinct alerts generated date wise

The number of distinct alerts generated per day is listed in Table 5.1. The alerts listed in the table include those alerts that have been triggered at least once on a particular day. The repetitive occurrence of the attack is not listed here. The data is from fourth and fifth week of DARPA IDS Evaluation dataset. So the results shown are from 29th march to 10th April. Traffic for one day *i.e.* 4th April is missing in the dataset for which no alerts were generated. A graph of this data is shown in Figure 5.1 which clearly shows that alerts generated by the anomaly + signature detection are more than individual signature detection approach on every particular day.

Table 5.1: Number of distinct alerts generated date wise

Date	Signature detection	Anomaly + signature detection
29 March	16	26
30 March	13	22
31 March	39	47
01 April	53	61
02 April	38	47
03 April	30	37
05 April	32	41
06 April	57	65
07 April	43	51
08 April	58	66
09 April	76	85
10 April	27	37

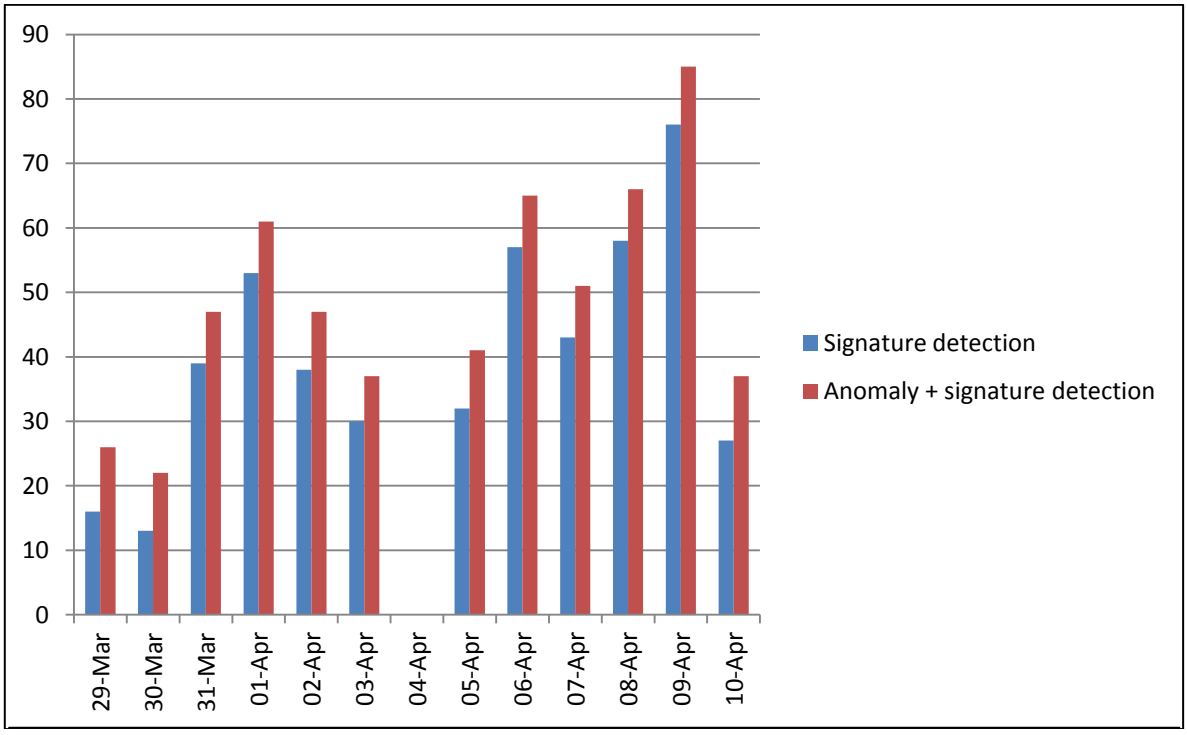


Figure 5.1: Number of distinct alerts generated date wise

The highest and lowest number of alerts generated by signature detection is 85 and 22 respectively. On the other hand anomaly + signature detection gives highest and lowest number of alerts as 76 and 13 respectively. The above figures clearly depicts that more alerts are generated in the combined approach.

ii) Number of alerts generated protocol wise

Another scenario that is used for comparison is the number of alerts generated by the packets containing protocols TCP and UDP. Table 5.2 and figure 5.2 show the result of the comparison.

Table 5.2: Number of alerts generated protocol wise

Protocol	Signature detection	Anomaly + signature detection
TCP	99	111
UDP	11	15

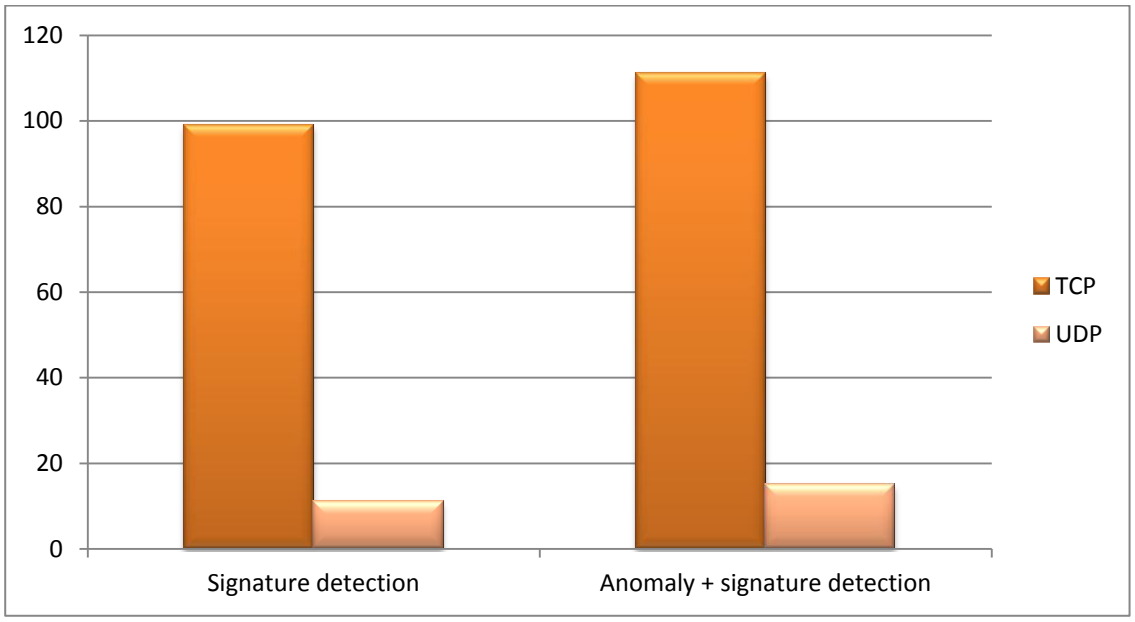


Figure 5.2: Number of alerts generated protocol wise

Out of the total number of attacks present in the TCP data packets, signature detection generates 99 alerts and anomaly + signature detection generates 111 alerts. Out of the total number of attacks present in UDP data packets, signature detection generates 11 alerts whereas anomaly + signature detection generates 15 alerts. For both cases of TCP and UDP, anomaly + signature detection performs better.

iii) Total number of attacks detected/ Detection rate

The performance of Intrusion Detection Systems can be measured in terms of number of attacks detected by it. The Intrusion detection rate denoted by δ is defined by

$$\delta = d/n$$

Where d is the no. of detected attacks and n is the no. of actual attacks.

The total number of alerts generated after testing both the approaches on fourth and fifth week of data comes out to be 116 in case of signature detection and 123 in case of anomaly + signature detection. This implies that signature detection detects 116 attacks out of total 201 present in the dataset. Anomaly + signature detection detects 123 attacks out of 201 attacks. Figure 5.3 shows the detection rate of the approaches. The detection rate is calculated as follows.

For signature detection,

$\delta = 116/201*100 = 57.71\%$

For anomaly + signature detection,

$\delta = 123/201*100 = 61.19\%$

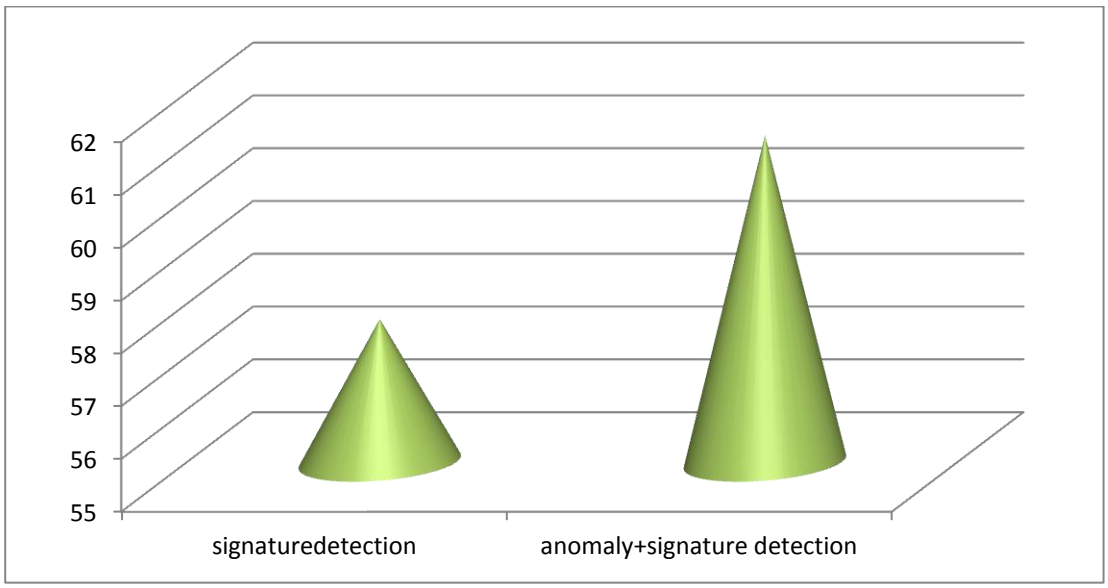


Figure 5.3 Detection rate

As evident from the above figure, the detection rate has considerably increased from about 57% to 61%. This implies that the hybrid approach of anomaly + signature detection is better than individual approach of signature detection.

5.2 Results and performance analysis of WEKA

This section discusses the output of WEKA. As explained in Chapter 4, a subset of large dataset having identical records but only different labels is used for analyzing it on WEKA. The subset of dataset consists of 23995 packets.

5.2.1 Performance metrics

Various performance metrics can be used to test the performance of IDS. The performance of IDS can be measured in terms of detection rate, accuracy, false alarm rate. The metrics measure the percentage of intrusions detected by the system and how many misclassifcations it makes.

Table 5.3: Standard metrics to evaluate Intrusions

	Predicted Normal	**Predicted Attack**
Actual Normal	True Negative(TN)	False Positive(FP)
Actual Attack	False Negative(FN)	True Positive(TP)

The performance metrics used are as follows.

- Correctly Classified Instances: It is sum of True Negative (TN) and True Positive (TP).
- Incorrectly Classified Instances: It is sum of False Positive (FP) and False Negative (FN).
- Accuracy: Accuracy can be defined as the proportion of correctly classified classes namely True Positive and True Negative over the total number of classifications as depicted in formula (5.1).

  Accuracy = (TN+TP/ (TN+TP+FN+FP)) * 100 ... *(5.1)*
- False alarm rate: False alarm rate is the proportion of normal data which is falsely detected and labelled as an attack, namely False Positive (FP) over the sum of False Positive (FP) and True Negative(TN) elements multiply by hundred. False alarm rate calculated based on formula (5.2).

  False alarm rate= FP/ (FP+TN)*100 ... *(5.2)*
- Detection rate: The total detected attacks among all the data is called detection rate. The detection rate is based on formula (5.3).

  Detection rate = TP/ (TP+FN) *100 ... *(5.3)*

5.2.2 Results and analysis

The experiment was run twice with JRip classifier. First run is on dataset containing the classes listed in Table 5.4 named as dataset (1) and second run is on dataset containing the classes listed in Table 5.6 named as dataset (2) in Chapter 4. A good classification result is achieved by JRip classifier as shown in Table 5.5 with only 13 false positives and 10 false negatives. The total number of correctly classified instances is 23972 and incorrectly classified instances is 23. The percentage of correctly classified instances is 99.9041% and the detection rate is 96.21%. This corresponds to high detection rate and low false alarm rate.

Table 5.4: Confusion matrix for dataset (1)

		Predicted class	
		Normal	Attack
Actual class	Normal	23718	13
	Attack	10	254

Table 5.5: Performance metrics and values – dataset (1)

S No.	**Performance Metric**	**Value**
1.	Total Number of Instances	23995
2.	Correctly Classified Instances	23972
3.	Incorrectly Classified Instances	23
4.	Accuracy	99.9041%
5.	False alarm rate	0.054%
6.	Detection rate	96.21%

The results of the second run of experiment is shown in Table 5.6 and Table 5.7. The classification result is achieved is shown in Table 5.7. The table points out that prediction is very good for classes WEB-MISC, WEB-CGI and Ftp telnet with 0 false negatives. The false positives are Prediction is reasonably good for classes Community and TCP small segment with 3 false negatives. The worst prediction is for class Http inspect with 5 false negatives and only one true positive. The prediction for Normal class shows 15 false positives. The total number of correctly classified instances are 23969 and incorrectly classified instances are 26. The percentage of correctly classified instances is 99.8916% and the detection rate is 95.83%.

Table 5.6: Confusion matrix for dataset (2)

		Predicted class						
		Normal	Community	Http inspect	WEB-MISC	WEB-CGI	TCP small segment	Ftp telnet
Actual class	Normal	23716	5	4	2	2	2	0
	Community	3	95	0	0	0	0	0
	Http inspect	5	0	1	0	0	0	0
	WEB-MISC	0	0	0	4	0	0	0
	WEB-CGI	0	0	0	0	5	0	0
	TCP small segment	3	0	0	0	0	144	0
	Ftp telnet	0	0	0	0	0	0	4

Table 5.7: Performance metrics and values – dataset (2)

S No.	**Performance metric**	**Value**
1	Total Number of Instances	23995
2	Correctly Classified Instances	23969
3	Incorrectly Classified Instances	26
4	Accuracy	99.8916%
5	False alarm rate	0.063%
6	Detection rate	95.83%

In intrusion detection, using classification accuracy as a performance metric is not adequate. In the above analysis, the dataset contains only 1.1% intrusion data and a large amount of normal data. This leads to JRip classifier achieving 99% accuracy in both the experiments. Hence better way of judging the performance is to use detection rate and false alarm rate as performance metrics. From the above results, it is clear that the classifier achieves good

detection rate and low false alarm rate. Also as the numbers of classes are increased in the data, accuracy of the classifier decreases, false alarm rate increases and detection rate decreases.

5.3 Summary of the chapter

As predicted from graphs, anomaly + signature detection approach shows better result in above parameters as compared individual approach. So merging of two approaches results in increased detection of attacks. Now a days integration is very common. It provides better performance which is proved by above results.

Chapter 6

Conclusion and Future Scope

6.1 Conclusion

The research carried out for the thesis mainly emphasize on the need to protect the information systems from various kinds of threats .The increase in the use of Internet and complex architecture of the networks has made the systems vulnerable to intrusions . The networks can be secured by deploying security solutions like Antivirus, Firewall, Honeypot and IDS. The objective of the thesis is to study techniques used by IDS for detecting intrusions. An intrusion detection system is used to monitor network traffic, check for suspicious activities and notifies the system or network administrator. There are mainly two techniques to detect intrusions. These are anomaly detection and signature detection. Snort is one of the most effective and widely used open source IDS which can be used to detect these threats. It is based on signature detection. Snort matches the incoming network traffic with the library of signatures called rule file to detect the attacks.

The advantage of Snort is that it can detect well known attacks and has low false alarm rate but it fails to detect new attacks. On the other hand, anomaly based IDS can detect new attacks. So both the techniques need to be combined so as to get better results. The goal of the thesis is to combine anomaly detection with signature detection. Snort is combined with anomaly detection based IDS called AnomalyDetection. DARPA IDS evaluation data set is used to test the performance of these systems. The results of signature detection are compared with anomaly + signature detection. The detection rate of hybrid approach is 61% and signature detection is 57%. From the experimental results, it is evident that the combined anomaly + signature detection approach is better than individual signature detection approach.

The results are further used to perform analysis on WEKA. The dataset that was used earlier to evaluate the performance of the systems is labeled. Labelling is done in two different ways so as to prepare two sample datasets. These datasets are tested on WEKA using JRip classifier. Testing on first dataset results in 13 false positives and 10 false

negatives which give the percentage of correctly classified instances to 99.8916%. Testing on second dataset results in 15 false positives and 11 false negatives which give the percentage of correctly classified instances to 99.9041%. However, the more realistic measures to analyze the performance of the classifier are detection rate and false alarm rate. As the numbers of classes are increased in the data, accuracy and detection rate of the classifier decreases and false alarm rate increases.

6.2 Future Scope

- This framework can be tested on live network traffic.
- Training data can be prepared from live network traffic. The training data can be used to train the classifier so that upon testing better results are obtained.
- Machine learning techniques complement practical Intrusion Detection Systems. Apart from measures used in evaluation such as accuracy, false alarm rate and detection rate, others measures can also be used.

References

1. B. H. Forouzan and D. Mukhopadhyay, *Cryptography and Network Security*. 2nd ed., New York: Tata McGraw-Hill, 2012.
2. W. Stallings, *Cryptography and Network Security Principles and Practices*. 4th ed., Prentice Hall, 2005.
3. K. Scarfone and P. Mell, ”Guide to Intrusion Detection and Prevention Systems (IDPS),” Gaithersburg, MD, Rep. NIST Special Publication 800-94,Feb. 2007.
4. D. Rozenblum, “Understanding Intrusion Detection System,” *www.sans.org/reading_room/whitepapers/detection/understanding-intrusion-detection-systems_337*, October 31, 2003.
5. S. Ioannidis *et al*., “Implementing a Distributed Firewall,” in *Proceedings of the ACM Computer and Communication Security (CCS)*, pp. 190-199, 2000.
6. “Honeypot Security”, *http://www.infosec.gov.hk/english/technical/files/honeypots.pdf*.
7. J. Levine *et al*., "The use of honeynets to detect exploited systems across large enterprise networks," *Information Assurance Workshop, Man and Cybernetics Society*, pp. 92-99, 2003.
8. A. Samrah, “Intrusion Detection Systems; Definition, Need and Challenges,” *http://www.sans.org/reading_room/whitepapers/detection/intrusion-detection-systems-definition-challenges_343*, October 31, 2003.
9. Lazarevic *et al*., "Intrusion detection: A survey," *Managing Cyber Threats*, vol. 5, pp. 19-78, 2005.
10. T. M. Wu, “Information Assurance tools Report- Intrusion Detection Systems,” 6th ed., Information Assurance Technology Analysis Center (IATAC), Herndon, VA, Sep.25, 2009.
11. J. Gómez *et al*., "Design of a snort-based hybrid intrusion detection system,” *Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living*, vol. 5518, pp. 515-522, 2009.
12. M. Roesch, “Snort- Lightweight Intrusion Detection for Networks,” in *Proceedings of 13th USENIX conference on System administration LISA '99*, CA, USA, Nov. 1999, pp. 229-238.

13. V. Paxson, “Bro: A System for Detecting Network Intruders in Real-Time,” in *Proceedings of 7th USENIX Security Symposium*, San Antonio, TX, 1999, pp. 2435-2463.
14. S. Northcutt, “SHADOW,” http://www.nswc.navy.mil/ISSEC/CID/, 1998.
15. U. Lindqvist and P.A. Porras, “Detecting Computer and Network Misuse through the Production-Based Expert System Toolset (P-BEST),” in *Proceedings of the IEEE Symposium on Security and Privacy*, May 1999.
16. K. Ilgun, “USTAT A Real-time Intrusion Detection System for UNIX,” M.S. Thesis, Univ. Of California, Santa Barbara, 1992.
17. G. Vigna and R.A. Kemmerer, “Netstat: A Network-Based Intrusion Detection Approach,” *Journal of Computer Security*, vol. 7, no. 1, pp. 37-71, 1999.
18. M. V. Mahoney and P. K. Chan, “PHAD: Packet Header Anomaly Detection for Identifying Hostile Network Traffic,” Dept. of Comput. Sci., Florida Inst. of Technology, Melbourne, FL, Tech. Rep. CS-2001-04, 2001.
19. M. V. Mahoney, “Network Traffic Anomaly Detection based on Packet bytes,” in *Proceedings of the ACM symposium on Applied computing*, March 2003, pp. 346-350.
20. M. V. Mahoney and P. K. Chan, “Learning Nonstationary Models of Normal Network Traffic for Detecting Novel Attacks,” in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, July 2002, pp. 376-385.
21. M. Szmit *et al*., "Traffic Anomaly Detection with Snort," Information Systems Architecture and Technology, Information Systems and Computer Communication Networks, Wydawnictwo Politechniki Wrocławskiej, Wrocław, pp. 181-187, 2007.
22. “Anomalydetection project page,” *http://www.anomalydetection.info*
23. D. Anderson *et al*., “Detecting Unusual Program Behavior Using the Statistical Component of the Next-Generation Intrusion Detection Expert System (NIDES), Computer Science Laboratory, SRI International, Menlo Park, CA, Tech. Rep. SRI-CSL-95-06, 1995.
24. P.A. Porras and P.G. Neumann, “EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances,” in *Proceedings of the 20th National Information Systems Security Conference*, Oct. 1997, pp. 353-365.
25. L. Ertoz *et al*., “The MINDS - Minnesota Intrusion Detection System,” *Data Mining: Next Generation Challenges and Future Directions*, Menlo Park: AAAI Press, 2004, ch. 11, pp. 199-218.

26. E. Eskin *et al*., "A Geometric Framework for Unsupervised Anomaly Detection: Detecting Intrusions in Unlabeled Data," in *Applications of Data Mining in Computer Security*, 2002, Springer, vol. 6, ch. 4, pp.77-101.
27. Sinclair," An Application of Machine Learning to Network Intrusion Detection," in *Proceedings 15th Annual Computer Security Applications Conference, (ACSAC'99), 1999,* pp. 371-377.
28. V. R. Vemuri, "Machine Learning in Intrusion detection," in *Enhancing Computer Security with Enhanced Technology*, Boca Raton: Auerbach Publications, 2006, ch. 5, sec. pp. 96.
29. G.V. Nadiammai *et al*., "A Comprehensive Analysis and study in Intrusion Detection System using Data Mining Techniques," *International Journal of Computer Applications,* vol. 35, no.8, Dec. 2011.
30. G. S. Oreku and F. J. Mtenzi*,"* Intrusion Detection Based on Data Mining," *in 8th IEEE International Conference on Dependable, Autonomic and Secure Computing*, 2009, pp.696-701.
31. Rajput *et al*. "J48 and JRIP Rules for E-Governance Data," *International Journal of Computer Science and Security (IJCSS),* vol. 5, no. 2, 2011.
32. W. Lee *et al*., "A data mining framework for building intrusion detection models," in *Proceedings of the 1999 IEEE Symposium on Security and Privacy*, 1999, pp. 120-132.
33. S. M. Lee *et al*., "A Hybrid Approach for Real-Time Network Intrusion Detection Systems," in *International Conference on Computational Intelligence and Security*, Harbin, China, Dec. 2007, pp. 712-715.
34. Kai Hwang *et al*., "Hybrid Intrusion Detection with Weighted Signature Generation over Anomalous Internet Episodes," *IEEE Trans. Dependable and Secure Computing,* vol. 4, no. 1, pp. 41-55, Jan.-March 2007.
35. J.E. Díaz-Verdejo *et al*., "Una aproximación basada en Snort para el desarrollo e implantación de IDS híbridos (A Snort-based approach for the development and deployment of hybrid IDS)," *IEEE Latin America Transactions*, vol. 5, no. 6, Oct. 2007, pp. 386–392.
36. J. Zhang *et al*., "Random-Forests-Based Network Intrusion Detection Systems," *IEEE Trans. Syst. Man Cybern. C, Appl. Rev.*, vol. 38, no. 5, pp. 349-359, Sep. 2008.

37. M. A. Aydin, *et al.*, "A hybrid intrusion detection system design for computer network security," *Computers & Electrical Engineering*, vol. 35, pp. 517-526, 2009.
38. J. Gómez, *et al*. "Design of a snort-based hybrid intrusion detection system," in *Proceedings of 10th International Work-Conference on Artificial Neural Networks (IWANN)* , Salamanca, June 2009. pp. 515-522.
39. Zhao *et al*., "Analysis and Design for Intrusion Detection System Based on Data Mining," in *8th International Workshop on Education Technology and Computer Science (ETCS),* March 2010, pp. 339-342.
40. J. Yang *et al*., "HIDS-DT: An Effective Hybrid Intrusion Detection System Based on Decision Tree," in *International Conference on Communications and Mobile Computing*, Shenzhen, vol. 1, April 2010, pp. 70-75.
41. Amza et al., "Hybrid Network Intrusion Detection," in *International Conference on Intelligent Computer Communication and Processing (ICCP),* Cluj-Napoca, Aug. 2011, pp. 503-510.
42. S. M. Hussein et al., "Evaluation Effectiveness of Hybrid IDS Using Snort with Naïve Bayes to Detect Attacks," in *2nd International Conference on Digital Information and Communication Technology and it's Applications (DICTAP),* Bangkok, May 2012, pp. 256-260.
43. K. Qazanfari et al., "A novel hybrid anomaly based intrusion detection method," in *6th International Symposium on Telecommunications (IST),* Tehran, Nov. 2012, pp. 942-947.
44. P. Jain and A. Sardana, " Defending against internet worms using honeyfarm," in *Proceedings of the CUBE International Information Technology Conference,* ACM, Sept. 2012, pp. 795-800.
45. G. V. Nadiammai and M. Hemalatha, "Anomaly Based Hybrid Intrusion Detection System for Identifying Network Traffic," *International Journal of Computer Science and Information Security*, vol. 10, no. 10, Oct. 2012.
46. R. U. Rehman, *Intrusion detection systems with Snort: advanced IDS techniques using Snort, Apache, MySQL, PHP, and ACID*. New Jersey: Prentice Hall Professional, 2003.
47. Snort Team, "Snort Users manual," http://www.snort.org/assets/166/snort_manual.pdf, May 23, 2012.
48. Sen, Soumya, "Performance Characterization & Improvement of Snort as an IDS," Technical Report, Princeton University, 2007.

49. M. Skowroński, and R. Wężyk , “Systemy Detekcji intruzów i aktywnej odpowiedzi (Intrusion Detection Systems and proactive response),” M. S. thesis, Technical University of Lodz , 2006.
50. “Security Information 2012,” International Conference, 9th ed., June 2012.
51. “The R Project for Statistical Computing,” http://www.r-project.org.
52. “DARPA intrusion detection evaluation,” http://www.ll.mit.edu/IST/ideval/data/data index.html
53. M. V. Mahoney, “A machine learning approach to detecting attacks by identifying anomalies in network traffic,” Ph.d. dissertation, College of Eng., Florida Inst. of Technology, Melbourne, FL, 2003.
54. K. Kendall,” A Database of Computer attacks for the evaluation of intrusion detection systems,” M. S. Thesis, Dept. of Elect. Eng. and Comput. Sci., MIT, June 1999.
55. M. V. Mahoney and P. K. Chan ,”An analysis of the 1999 DARPA/Lincoln Laboratory evaluation data for network anomaly detection,” in *Proceedings of 6th International Symposium*, Pittsburgh, PA, Sept. 2003, pp. 220-237.
56. S. T. Brugger and J. Chow, “An assessment of the DARPA IDS evaluation dataset using Snort,” Tech. Rep. CSE-2007-1, Nov. 2005.
57. J. McHugh, “Testing Intrusion Detection Systems: A Critique of the 1998 and 1999 DARPA IDS evaluations as performed by Lincoln Laboratory,” *ACM Transactions on Information and System Security*, vol.3, no.4, pp. 262–294, Nov.2000.
58. “Wireshark, Man Pages,” http://www.wireshark.org/docs/manpages/wireshak.html.
59. M. Hall *et al.*, ”The WEKA Data Mining Software: An Update”, SIGKDD Explorations, Volume 11, no. 1, 2009.
60. Gullett, “Snort 2.9.1 and Snort Report 1.3.2 on Ubuntu 10.04 LTS installation guide,” http://www.symmetrixtech.com, Sept. 2011.

List of Publications

1. Tejvir Kaur and Sanmeet Kaur**, "**Comparative Analysis of Anomaly Based and Signature Based Intrusion Detection Systems Using PHAD and Snort," poster presented at Security and Privacy Symposium, February-March 2013, Indian Institute of Technology, Kanpur.
2. Tejvir Kaur and Sanmeet Kaur, "Evaluation of Rule based Machine learning algorithms on NSL-KDD dataset," International journal of Computational Intelligence and Information Security, vol 4, no. 6, pp. 42-49, June 2013.