Chapter I

Introduction

1.1 Introduction

As a result of the technological advances in recent years, we have become increasingly dependent on global networks when engaging in social, business, and educational activities. With the explosive use of computer networks, a number of security issues on the internet and in computer systems have been raised.

In the dawn of the information age, information has been attributed with an utmost important asset of an organization .Especially, organization of financial, product trading and data services via internet system need the highest security for customer's data, if those data are stolen by the people who have malicious intention, it will cause high damage to the business and its effect to qualities to the provided services or the system was become unavailable to provide the services that is requested from it's. Moreover, the network communication technology is the fast improved and more complicated noun. If the organization doesn't have an efficient security mechanism for protecting data and network it makes the attackers to easily find vulnerabilities to cause malicious activities on the networks.

The using anti-threat applications such as antivirus software, firewalls and spyware-detection programs are became not enough to detect and prevent the latest malicious activates that resulting from advance tools and techniques that are used by the intruders that make the needs for a technologies that help to detect and prevent the unauthorized traffic that passed through a good intrusion detection with hybrid approaches [1].

Intrusion detection and prevention is the new technology that monitors the activities on the network or on the specific devices like a servers to detect and prevent unauthorized traffics. There are many types of intrusion detection techniques that can be added to devices such as signature based and anomaly based intrusion detection that full-filled to intrusion detection system such as snort system.

Snort is a free open source network intrusion detection and prevention system, has capable of performing real-time traffic analysis and packet logging on IP networks. Also , It can perform packets capturing and analysis for many types of protocols, content searching/matching, and can be used to detect a variety of attacks and like flooding attacks .In additional to that it support two types of approaches signature based and anomaly based detection techniques [2].

1.2 Problem Statements

1.2.1 Background

In the last decades, cyber-attacks have had a significant impact on the security of businesses and organizations that resulting from rapid progression of computer technology, computer violations are increasing at a fast pace. Such malevolent activities become more and more sophisticated and can easily cause millions of dollar in damage to an organization.

Through nay 2017, organizations have awarded hackers over $17 million in bounties on Hacker One, and over $7 million awarded in 2016 alone. The detection of these attacks is the most important issues for any organization and it can be done by the using of two approach signature based and anomaly based detection for the detection of attacks.

The signature base based on compared the observed data with predefined rules if it's match mean that there is types of attacks but the anomaly base detection compare the observed behavior with the behavior of normal user if it's different means that this behavior of attacks. During the past year the devolved intrusion detection systems have many shortages for detecting attacks and produce unacceptable results of detection accuracy rate and false alarm rate.

1.2.2 The problem

The increasing number of cyber threats likes flooding attacks in the networks of the organization and there were various approaches to intrusion detection are currently being used, but they are relatively ineffective for detecting flooding attacks and it will cause many problems such as:

1. May detect and prevent only known or unknown attacks.
2. Making a resource (e.g. CPU, memory, bandwidth, disk space) unavailable to its legitimate users, by exhausting it
3. Bring a network or service down by flooding it with large amounts of traffic **[**3**].**
4. The detection rate nearly is less than or equal to 95.5% and 1.8% for false

positive rate, but in nowadays these values are not satisfied.

1.3 Importance of Research

The proposed hybrid intrusion detection system that based on signature based and anomaly detection that based on decision tree has ability to monitor the activities in the network to detect flooding based attacks and it help the employees to do their task more quickly. Also, It make the customers have guarantee to take few response time to finish from required services without denial of their services.

1.4 Hypothesis

Using of hybrid signature based and anomaly based detection model based on decision tree that based on c4.5 algorithm a of data mining techniques that applied on the proposed hybrid IDS on snort engine has capable of real-time intrusion detection and network security monitoring for malicious activities or unauthorized traffics such as flooded messages .Also its offer several advantages over alternative systems and can represented it's in:

i. Providing higher security.

ii.It's support high availability and stability.

iii. Producing better detecting results in terms of normal and abnormal behaviors of captured packets than the existing results of currently used system.

1.5 Objectives of the Research

The research aim to:

1. Collect dataset about the normal behaviors for valid users.
2. Build secure and robust hybrid IDS/IPS based on signature and anomaly detection based on decision tree and neural network to provide high detection rate and low false alarm.
3. Demonstrate how some types of distributed denial of services (DDOS) attacks or flooding attacks that will occur and implement the behaviors of them through various types of flooding messages.
4. Analysis the performance of proposed IDS/IPS system.
5. Evaluate the proposed of hybrid intrusion detection system.

1.6 Methodology

For this research, will trying to find a method that provides the best way to detect malicious activities and to solve the shortage of existing instruction detection system and this done by build enhanced hybrid instruction detection system that based on signature technique and anomaly based technique by using decision tree of data mining classification techniques to make deep packets inspection that help in detecting of known and unknown attacks and providing better results of detection accuracy rate and false alarm rate than existing results.

1.7 Boundaries

The proposed intrusion detection system was based on linux operating system (Ubuntu, Kali) through using the signature and anomaly detection that is based on decision tree of data mining technique under using snort and weka to detect some types of flooding attacks such as: ICMP flood, TCP flood, UDP flood, SYS flood and HTTP flood and will use iptabel firewall for prevention them.

1.8 Contents of Research

Chapter one is introduction, this chapter gives introduction about the project, defining the problems, important, proposed solution, objectives and boundaries. Chapter two is literature review, it's consists of two parts. Part one represents a general background about the sending and receiving data or the observed data and approaches of detection. Part two is the related studies and security techniques that used in the organization for detecting flooding attacks.

Chapter three also contains two parts, First part explains the tools and techniques that is used in this project, and the second part is the project methodology that will be taken and will represented it by drawing graphs that illustrate the project functionality.

Chapter four is implementation and results analysis, this chapter conation implementation for the proposed hybrid intrusion detection system and analysis the results. Chapter five is the conclusion and future works that contains the results of the project and recommendations for the future works.

Chapter II

Literature Review and Related Works

2.1 Background

This chapter will discuss about the data that are transmitted on the network and the important of quality of service for the requested services from the servers and the types and technique of instruction detection system, also it contains the literature review for related studies with their security techniques that were used for intrusion detection.

The computer networks and internet play important role in our life and there are daily interactivity with applications to do their tasks inform of transaction, banking, shopping through the internet. As rapid growth of these applications there are many new threats or malicious activities that effects to the resources or the performance on the network that cause denial of services to the services that are provided to the customers, so that the security for detecting unauthorized activities are become very important on the networks

The unauthorized activities into a computer system or network are one of the most serious threats to computer security that effect to the resources on the networks so that will need more accurate intrusion detection system to maintains the confidentiality , integrity and availability of security features on the networks.

Intrusion as term that can be defined as any unauthorized activities on the computers network or on the internet with the regardless the types of intrusion approaches or their methods that are flows .The need of the efficient system called intrusion detection system(IDS), to detect attacks will become the most issues today [4].

2.2 Intrusion Detection System

Intrusion Detection System (IDS) is meant to be a software application which monitors the network or system activities and finds if any malicious operations occur and this is done through using different types of intrusion detection approaches, techniques, methods and algorithms. If the intrusion is detected more quickly , it enable the network administrator to early identify the type of intrusion and select the appropriate protection mechanisms that prevent the intruders from effecting to the systems , resources and from causing any damages to the systems , so that we must select the appropriate types of IDS location ,techniques and deployment methodology [5].

2.2.1 Deploying Intrusion Detection Systems (IDS's)

Intrusion detection technology is a necessary addition to every large organization's computer network security infrastructure to detect the malicious activities. However, given the deficiencies of today intrusion detection products, and the limited security skill level of many system administrators that effect to the development of IDS, an effective IDS deployment requires careful planning, preparation, prototyping, testing, and specialized training.

2.2.2 Types of Deploying IDS's

1. Network Based Intrusion Detection System (NIDS)

In this type of IDS the system sensor was located behind the external firewall or It is most specially denoted at a boundary between networks such as in routers, firewalls, virtual private networks to detect the attacks that originating for outside worlds and try to penetrate the network's perimeter defense [6].

The internal host systems are protected by an additional NIDS to mitigate exposure to internal compromise. The use of multiple NIDS within a network is an example of a defense-in-depth security architecture to create more than one level of the detection, such as represented in figure 2.1:

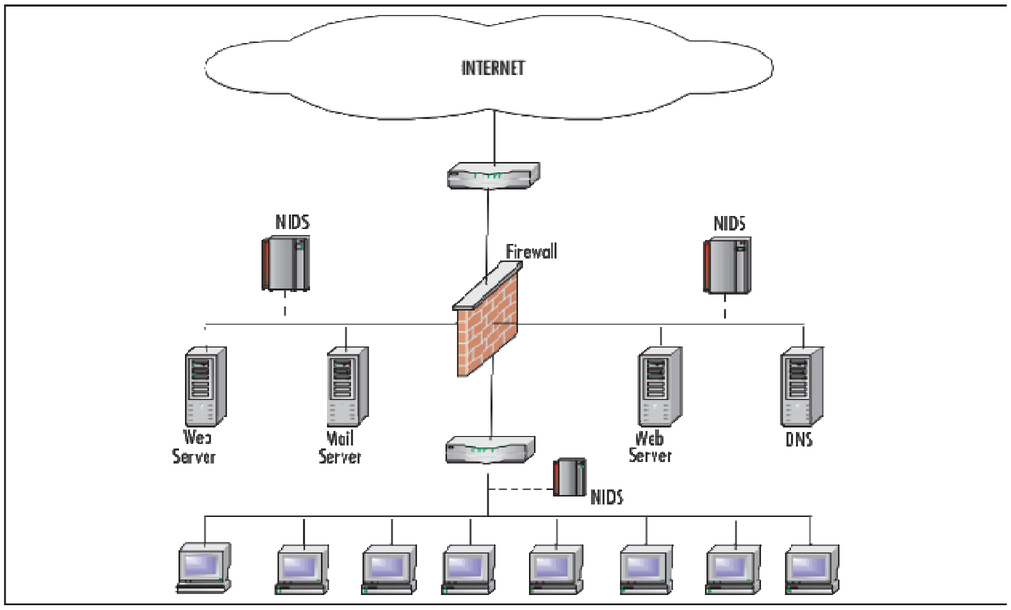


Figure 2.1: NIDS Network [6]

The main disadvantage of this type of intrusion detection system is that it has a single point of failure. Moreover, it is weak against denial of service attacks. It monitors the whole network and denoted at the boundary of the network. But it is not suitable for securing each of the hosts within the network. If an intruder can bypass it, all the systems within the network would be in troubling the problem so that it must configured it with the good security mechanisms [7].

2.Host Based Intrusion Detection System (HIDS)

Host-based IDS (HIDS) technology, software engineer are installed on each of the computer hosts of the network to monitor the events occurring within that host only not the whole network as it see in figure 2.2:

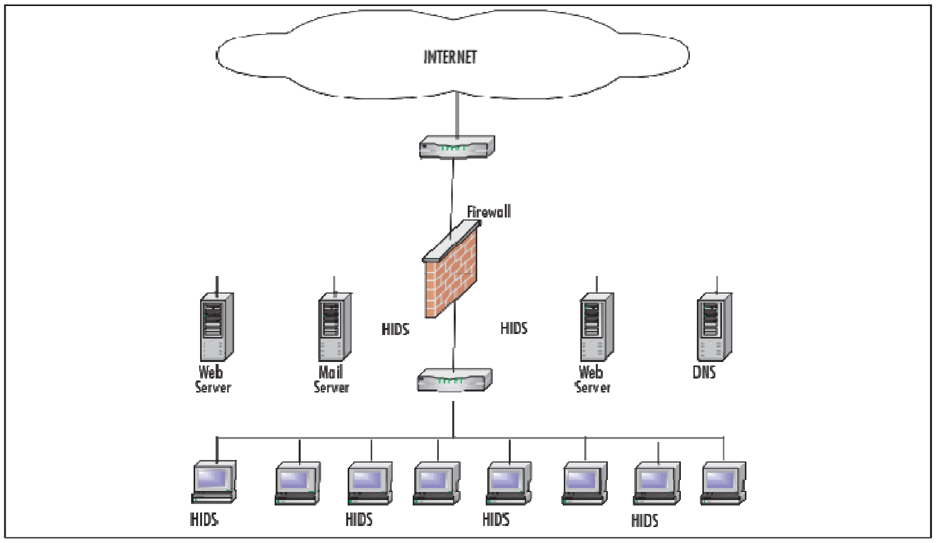


Figure 2.2: HIDS Network [7]

HIDS notice network traffic and system-specific settings such as software calls, local security policy, local log audits. For analysis they use host system's logging and other information or action that occur on the system and registered on log file and it provide these advantages:

1. Monitors system activities and can detect attacks that network IDS fail to detect them without require additional hardware
2. It provide near real time detection for an attacks that success or failure for an attacks to the system [8][9].

3. Wireless Intrusion Detection System

A wireless local area network intrusion detection system is similar to NIDS in that it can examine wireless network traffics and analyze them to identify which external users that trying to connect to our access point (AP) to cause any types of malicious activities .This type of IDS are developed into access points or in wireless routers or behind the firewall [10]..

Generally, the efficiency of detection attacks depends on the types of detection techniques that full-filled on the intrusion detection systems.

2.3 Techniques of Intrusion Detection System

The most commonly types of techniques that used by intrusion detection systems for detecting intruders are misuse detection (also called signature based detection) and anomaly detection (Behavior Detection).

2.3.1 Signature Based Detection

The signature is a pattern that corresponds to a known threat. It's also known as knowledge based detection or misuse detection In signature based detection, observed events are compared against the pre-defined signatures in order to identify possible unwanted traffics if it's match with the predefined rules or knowledge that defined in IDS system as a represented this operations in figure 2.3:

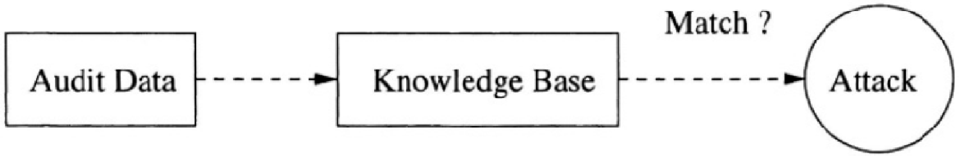


Figure 2.3: Typical Knowledge-Based IDS [8]

This type of detection technique is very fast and easy to configure. It's is very effective at detecting known threats but largely ineffective at detecting previously unknown attacks that use new techniques or different approaches that commonly unknown so that to detect these types of attacks will require to update the detection rules of IDS. Still some organizations use signature base approach by taking it limited capabilities of it's and may be a curate for detecting the attacks that attacks them [10].

2.3.2 Anomaly Based Detection

Anomaly based detection is another approach of intrusion detection techniques. It was inverted to solve the shortages of signature based detection and provides many features for detecting unknown attacks.

In this types of 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. For instance, anomaly based IDS will detect that an Internet protocol (IP) packet is malformed. It does not detect that it is malformed in a specific way, but indicates that it is anomalies. 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 [11].

Then the IDPS will compares the characteristics of current activity to thresholds related to the profile if it's anomalies mean that there are abnormal activities of attacks as it show this process in figure 2.4:

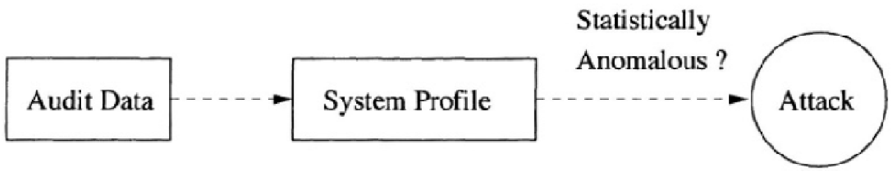


Figure 2.4 : Typical Anomaly-Based IDS [8]

.One of the advantages of using a anomaly based detection approach, it can be very effective at detecting previously unknown threats, but the common problems with anomaly-based detection when including malicious activity within a profile, establishing profiles that are not sufficiently complex to reflect real-world computing activity, and generating many false positives alarms [12].

2.3.3 Statefull Protocol Inspection

Statefull protocol inspection is similar to anomaly based detection, but it can also analyze traffic at the network and transport layer and vended-specific traffic at the application layer, which anomaly-based detection cannot do. It's compares predetermined profiles of generally accepted definitions of benign protocol activity for each protocol state against observed events to identify deviations [10].

The common Problems with stateful protocol analysis include that it is often very difficult or impossible to develop completely accurate models of protocols, it is very resource-intensive, and it cannot detect attacks that do not violate the characteristics of generally acceptable protocol behavior [13].

Nowadays the using of only one types of intrusion detection techniques was became inefficient to detect unauthorized activities , so the need of developing hybrid intrusion detection systems with the good methods of intrusion detection techniques was become the most issues.

2.3.4 Hybrid Intrusion Detection Systems (HIDS)

An IDS that use a mix of both techniques of intrusion detection such as signature based and anomaly based detection techniques to obtain the features of both techniques for detection the malicious activities with high accuracy and detection rates with low false alarm rate [14]

The anomaly based detection technique need method that have specific algorithm that used to detect anomalies. The decision tree of data mining techniques will selected it as appropriate technique that provide method that help to detect the unauthorized activities or attacks .

2.4 Data Mining Techniques

Data mining (DM) is the process of automatically searching large volumes of data for patterns using association rules to extract information that help to obtain or build knowledge that can used to solve problems in particular domains by using data mining techniques. DM is also called Knowledge-Discovery and it's provides many techniques with their own advantages and features that can use this technique used in many fields such as for intrusion detection to help in finding and identifying the normal and abnormal activities according to predefined thresholds [15].

2.4.1 Most common data mining techniques

I. **Classification:** It's attempt to build predefined classification model for classifying attacks or the objects according to particular class [16].

II. **Clustering**: it’s aim to define a set of cluster and each cluster contain data or object that like each other to enable to analyze and identify the similarity and dissimilarity between the objects.

III. **Association**: Describes relationships within tuples. Detection of irregularities may occur when many tuples exhibit previously unseen relationships. It can be used for general analysis similar to categorization, or for detecting outliers that may or may not represent attacks [16].

The main function of the model that we are interested in is classification, as normal, or malicious, or as a particular type of attack and it provide high accuracy with the different rates it depend on the types of techniques that provides methods that are used .The common representations for data mining techniques include rules, decision trees, linear and non-linear functions [15][16].

2.5 Data Mining Classification Techniques

Classification is the process of finding a set of models (or functions) which describe and distinguish data classes or concepts, for the purposes of being able to use the model to predict the class of objects whose class label is unknown. The derived or classification model is based on the analysis of a set of training data (i.e., data objects whose class label is known).

The classification model may be represented in various forms, such as classification (IF-THEN) rules, decision trees, mathematical formula, or neural networks [16].

2.5.1 Decision Trees:

A decision tree is the data mining method technique that can be represented as form of tree structure, where each node denotes a test on an attribute value, each branch represents an outcome of the test, and tree leaves represent classes or class distributions. The decision tree consists of nodes that form a rooted tree, meaning it is a directed tree with a node called a “root” that has no incoming edges and it’s a parent node to all other nodes. All other nodes have exactly one incoming edge. A node with outgoing edges is referred to as an “internal” or “test” node. All other nodes are called “leaves” (also known by decision nodes) [16].

Naturally, decision tree can be easily converted to classification rules and the complexity of tree has a crucial effect on its accuracy. The tree complexity is explicitly controlled by the stopping criteria used and the pruning method employed of particular algorithm.

The most Common Algorithms for Decision Tree are:

I. ID3

"The ID3 algorithm is considered to be a very simple decision tree algorithm [Quinlan (1986)]. Using information gain as splitting criteria, the ID3 ceases to grow when all instances belong to a single value of a target feature or when best information gain is not greater than zero. ID3 does not apply any pruning procedure nor does it handle numeric attributes or missing values "[16].

II. C4.5

"C4.5 an evolution of ID3, presented by the same author [Quinlan (1993)], uses gain ratio as splitting criteria. The splitting ceases when the number of instances to be split is below a certain threshold. Error-based pruning is performed after the growing phase. C4.5 can handle numeric attributes. It can also induce from a training set that incorporates missing values by using corrected gain ratio criteria as presented above "[16].

III. CART

" CART stands for Classification and Regression Trees. It was developed by [Breiman et al. (1984)] and is characterized by the fact that it constructs binary trees, namely each internal node has exactly two outgoing edges. The splits are selected using the towing Criteria and the obtained tree is pruned by Cost-Complexity Pruning. When provided, CART can consider misclassification costs in the tree induction. It also enables users to provide prior probability distribution" [17].

2.6 Related Works:

In the recent years, various hybrid IDS systems have been developed to achieve the best possible performance. In this section, we will review some of these methods

P.Akshaya [18] has developed intrusion detection system based on genetic algorithm and neural network to take the advantages of classification abilities of genetic algorithm and neural network for intrusion detection system to detect new attacks with high detection rate and low false negative and it used the KDD99 benchmark dataset and obtained reasonable detection rate. The result of classification rate for experiment result for known attacks were 80%, and for unknown attacks were 60%

L. Khalvati , M. Keshtgary and N. Rikhtegar [19] have proposed intrusion detection system that is based on novel hybrid learning approach that combine the K-Medoids clustering and Selecting Feature using the support vector machine method to provide a better accuracy and detection rate and also false alarm rate than the others. The experimental results on the KDDCUP'99 dataset have shown that this method is capable of achieving 91.5 % for the accuracy, 90.1% for detection rate and 6.36 for False alarm rate for detecting DOS attacks.

Pradhnya Kamble, R. C. Roychaudhary [20] have proposed hybrid approach for intrusion detection that based on using two a new learning methodology towards developing a novel intrusion detection system (IDS) by Back propagation neural networks (BPN) and Extreme Learning Machine(ELM). Specifically, this paper proposes the application of an Extreme Learning Machine based approach to the network-based intrusion detection system (IDSs). Good performance is achieved and preliminary results are reported in this paper.

Özge C epheli1Saliha Büyükçorak, and Güneg Karabulut Kurt [21] have build hybrid instruction detection system that based on signature based and anomaly based IDS for detecting DDoS under using expectation maximization algorithm. The results show that the proposed hybrid model of intrusion detection system has a 92.1% TPR and 1.8% FPR, and with signature-based detector we get 64.7% TPR and 13.2% FPR.

Aliya Ahmad , Bhanu Pratap Singh Senga [22] were developed instruction detection system based on support vector machine using BAT Algorithm to identify all intrusion correctly and minimized the false alarm generation to increase the performance of the system. The experimental results have shown that the proposed IDS had accuracy reaches up to 94.309% with minimum FPR and FNR whereas classification rate for Existing Method (IG-ABC SVM) is 89.799%.

Table 2.1: Comparison Table

No	Paper Name	Author	Techniques	Result	Open Issues
1	Intrusion Detection System Using Machine Learning	Akshaya a	Genetic algorithm and neural network based on	They Proposed intrusion detection model has ability to recognize an attack, to differentiate one attack from with result for	1.The suggested model will became ineffective in nowadays because it has 60 80% for detection rates and the

	Approach [18]		machine learning	known attacks were 80%, and for unknown attacks were 60% .	enhancement is required 2. The proposed system does not has ability to detect known attacks. 3. There are several method that can increase detection rate and decrease the false alarm.
2	Intrusion Detection Based on a Novel Hybrid Learning Approach [19]	L. Khalvati , M. Keshtgary and N. Rikhtegar	K-Medoids clustering and SVM algorithms	They have proposed IDS that is based on novel hybrid learning approach that has capable of achieving 91.5 % for the accuracy, 90.1% for detection rate and 6.36 for False alarm rate for detecting DOS attacks.	1.The clustering approach that it used has low a curacy if they use the classification technique we obtain better results. 2. It can detect unknown attack only with unsatisfied values of accuracy, detection rate and false alarm rate.
3	Discrimination Prevention in Data Mining for Intrusion and Crime Detection [20]	Pradhnya Kamble, R. C. Roychaudhary	Back propag -ation neural networks and Extreme Learning Machine	Proposed hybrid approach for intrusion detection that based on using two a new learning methodology towards developing a novel IDS system.	The proposed system is not easy to configure and it require more training time but it has self adaptive learning

NO	Paper Name	Author	Techniques	Results	Open Issues
4	Intrusion Detection System Based on Support Vector Machine Using BAT Algorithm [21]	Özge C epheli1 Saliha Büyükç orak,and GüneG Karabul ut Kurt	Signature based and Anomaly based under using EM algorithm	Build signature based and anomaly based IDS for detecting DDoS that has a 92.1% TPR and 1.8% FPR, and with signature-based detector we get 64.7% TPR and 13.2% FPR.	1.The produced model can able to detect know and unknown attacks with high value of TPR and FPR and them need to decrease 2.They doesn't use any effective protection mechanism
5	Intrusion Detection System Based on Support Vector Machine Using BAT Algorithm [22]	Aliya Ahmad , Bhanu Pratap Singh Senga	Support Vector machine using BAT Algorithm	Proposed induction detection system that improves the detection accuracy and reduces false alarm rate with accuracy reaches up to 94.3%	1.It enhanced the accuracy to 94.3 for detecting know attack but it doesn't detect known attacks. 2. It requires good preprocessed dataset to work well.

2.6.1 Evaluation of literature review and related works

From reading the related studies, I found that the most intrusion detection systems that was proposed have many shortages such as them have unsatisfied results of accuracy of detection rate and false positive rate for the detection of the attacks in nowadays and it can detect known or unknown attacks so that will need to enhanced it by building hybrid IDS/IPS system to provide better results than existing systems and this is the main goals of this thesis (see table 4.1).

Chapter III

Methodology

3.1 Introduction:

This chapter mainly contains the methodology and definition of require tools for developing the proposed instruction detection system. The project aim to build efficient hybrid intrusion detection system that is based on signature based detection and anomaly detection based on decision tree and neural network to detect the flooding attacks with high accuracy and detection rates.

Today intrusion detection system is still in infancy and need lot of research work to be done to make the intrusion detection even more successful. There are a huge number of issues and challenges in current intrusion detection system which needs the immediate and strong research attention to make enchantment for it's.

The issues and challenges are as:

1. Deficiency or incomplete Data set:

The updated data set has high effective for accuracy of detection and the cleaning of the data set are very important for building the model of detection.

2. The signature based intrusion detection system detects only known attacks.

3. Detection Algorithms:

The IDS need efficient algorithm that enable to detection of attacks with high accuracy of detection rates that implement under the appropriate tools.

3.2 The required tools:

3.2.1 Ubuntu:

Ubuntu is a complete linux operating system, freely available with friendly user interfaces that support many types of languages to make easily interaction with the software s on it's and it make the people should have the freedom to customize and alter their software in whatever way they see fit and it is suitable for both desktop and server use with the version 17.10 and it's has many features that provide it's [23].

Features of using ubuntu:

1. Friendly user interface and it also support the command line interface with the upgraded version of the kernel.

2. Fast, secure operating system and with thousands of apps that available for download.

3.2.2 Snort:

Snort is an open-source; free and lightweight network intrusion detection system (NIDS) software created by martin roesch in 1998 for linux and windows to detect to the malicious acclivities that was occurred

Snort is a packet sniffer that monitors network traffic in real time to capture and analyze each packet closely to detect a dangerous payload or suspicious anomalies.Snort is currently the most popular free network intrusion detection software , it used in building the proposed hybrid IDS because it provide the following features :

1. Real time intrusion detection: it help to detect attacks more quickly and on real time with the using predefined rules and anomalies.

Snort rules are easy to write and it’s greater accuracy

2. High adaptability and its support three mode that can be used to solve your own network security problems.

Snort Modes:

Snort operates in three basic modes: packet sniffer, network intrusion detection mode and packet logger mode.

1. Packet Sniffer Mode

In packet sniffer mode the snort can be used as packet sniffer to capture the packets and analyze them and can also log these packets to a log file or into the database. The file can be viewed later on [24].

2. Network Intrusion Detection (NID) Mode

In network Intrusion Detection mode, Snort logs only those packets which match a certain rule (pre-defined attack signatures) and generates alarms. Common rules (signatures) can be obtained from the installation files it’s self, and new rules keep updating regularly [25].

3. Packet logger mode

This mode is used to log a packet in log file.

3.2.3 Wireshark

Wireshark is an open source tool that is used for packets capturing and analyzing. It's referred to it's as a packet sniffers or network analyzer. It can be used to examine the details of traffic at a variety of levels to provide information that is needed by the network administrators such as: date and time of capture of packet, source ip address, source port number, destination ip address, destination port,…. etc [26**].**

Advantages of using Wireshark:

a. Network administrators use it to troubleshoot network problems.

b.Network security engineers use it to examine security problems.

c,Using it to verify network applications

3.2.4 Weka:

Weka (Waikato Environment for Knowledge Analysis) is tool that contains a set or collection of machine learning algorithms for data mining tasks. The algorithms can either be applied directly to a dataset or called from your own Java code. Weka contains tools for data classification, clustering, regression, association rules, and visualization. It is written in java, developed at the university of Waikato in new zealand [27].

Advantages of using Weka:

1. It's available free and it's open source tool.
2. Its portable software and cam run on any platforms.
3. Ease of use due to it has friendly user interfaces
4. It's contain a set of techniques that can be use for data preprocessing , transformation and building model that help to discovers knowledge from large dataset.

The proposed method of data mining techniques are applied it on the network security laboratory knowledge discovery data mining (NSL-KDD) dataset with 41 features and generally it consist from 42 feature one of them is used as the class.

3.2.5 Dataset Description

The NSL-KDD data set is the updated version of the KDD cup99 data set .It's can be used by many types researchers to analysis it for solving many problems by employing different techniques and tools with a universal objective to develop an effective intrusion detection system. It’s stand for network security laboratory knowledge discovery data mining [28].

3.3 Project Methodology

The collected data-set or observed traffics are need to prepossessed to help the hybrid IDS techniques to detect attacks.

After that the observed traffic are passed to hybrid IDS (Signature and Anomaly) detection system as seen in figure 3.1. In signature based detection, The detector(Engine) will compare the observed traffics against the pre-defined signatures in order to identify possible unwanted traffic after the observed traffic are matches with the pre-defined signatures then it will generate attacks alert messages to the administrator as shown in the top part in the below figure 3.1:

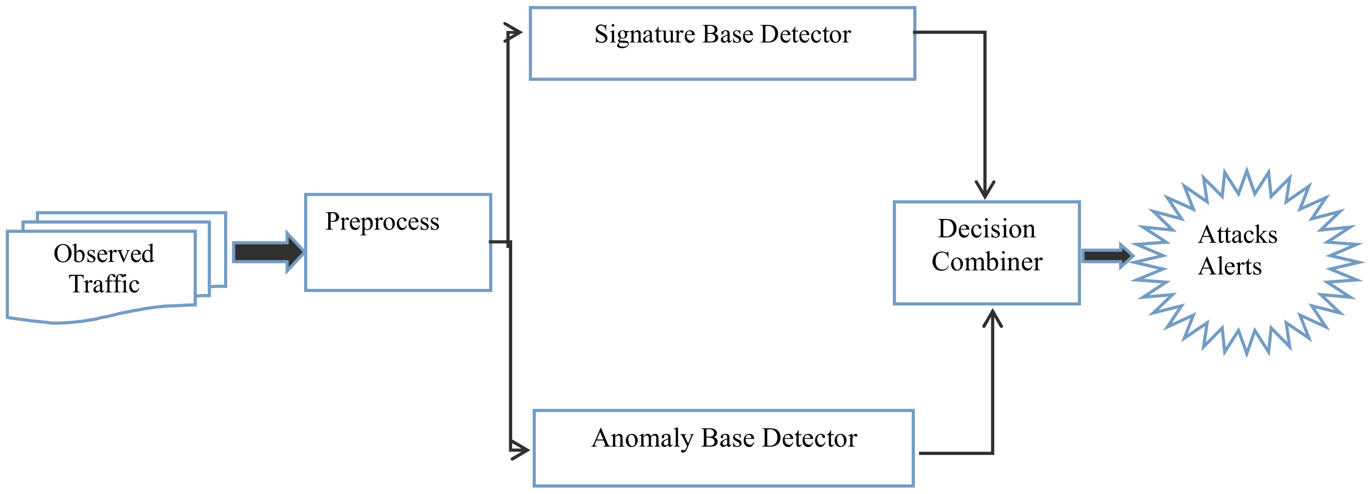


Figure 3.1: Architecture of the proposed Hybrid IDS

From the above figure 3.1, In Anomaly based detection engine, the anomaly base detector must receive processed dataset so that will need to preprocess it by extracting the important features (e.g . duration , protocol, service ..,etc) and cleaning data-set from any noise or irrelevant values then will reduce the features by remove similarity attributes after that will be able to transfer it to required format that will needed by the methods of the proposed hybrid intrusion system and there is no significant differences between preprocess in figure 3.1 and 3.2 but the figure 3.2 illustrate operation of anomaly based model with the more details and that it’s seek to define it.

The removing of irrelevant features from the dataset can be done by using Info gain feature selection technique and it help to develop a robust classifier that will be computationally efficient and effective techniques for cleaning data-sets as represented steps in figure 3.2:

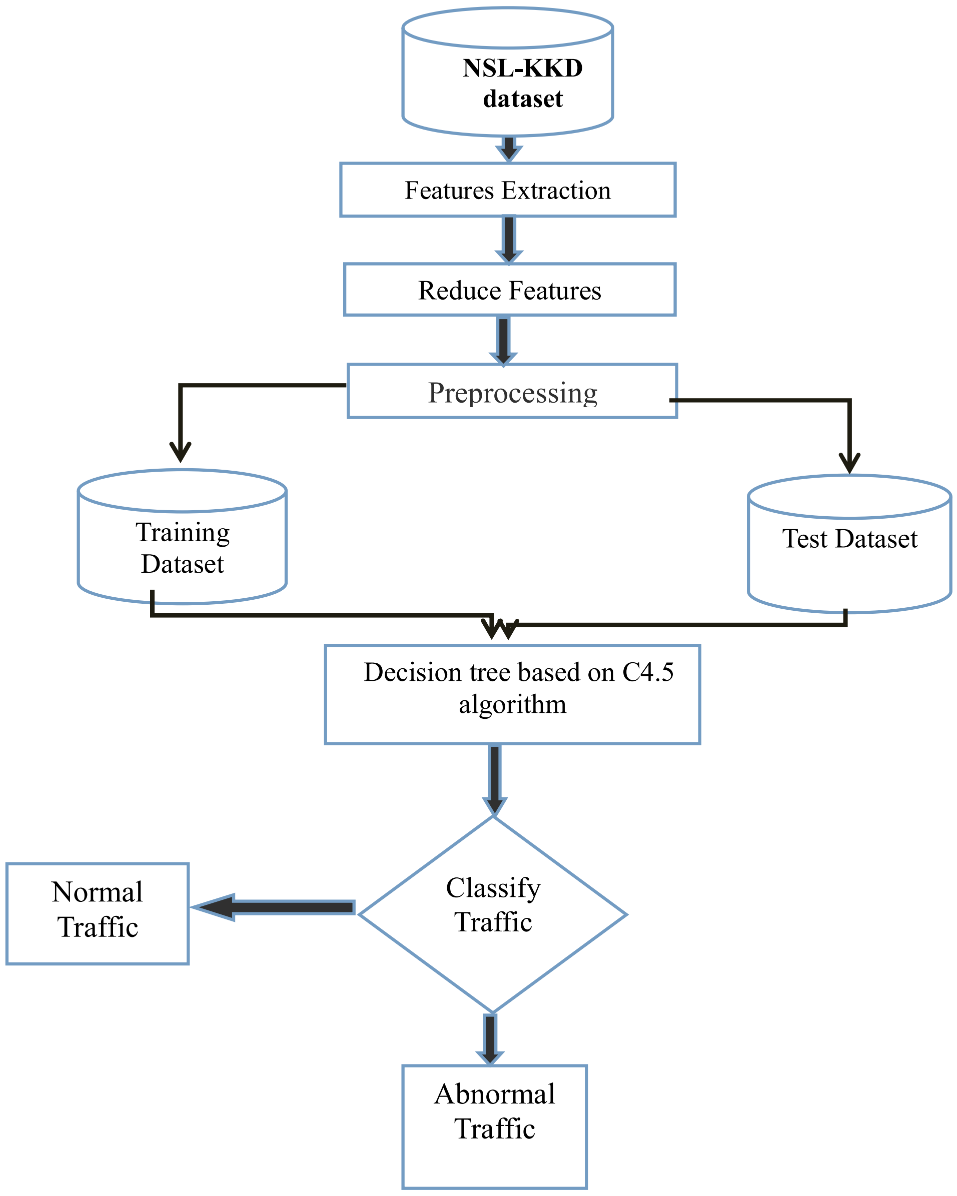


Figure 3.2: Steps of building proposed Anomaly detection model

After that , the prepossessed dataset are passed to the classifier of the proposed intrusion detection system that work with hybrid techniques as it saw in figure 3.1, then the training dataset are passed to classifier to classify it as normal and abnormal traffics according to predefined classification as it’s represented in figure 3.2.

As it saw in figure 3.1, the hybrid techniques with their methods will work together by using decision combiner to work as logical module that generate decision for detecting attacks based on signature and anomaly detection technique to provide high accuracy of detection rates to help administrators to choose and configure the appropriate protection mechanism.

3.3.1 Reasons of using classification techniques

Because the **c**lassification techniques are useful to handle large amount of data. Classification is used to predict categorical class labels. Classification models are used for classifying newly available data into a class label.

3.3.2 Reasons of using decision tree based on C4.5

Because it's divides the classification problem into sub-problems. It builds a decision tree which in turn is used to develop a model that is used for the classification purpose. The decision tree provides these features:

1. Decision trees can be very fast and power efficient.
2. Decision trees have an easy to follow natural flow from visual representations of the data so that it's easy to understand.
3. They are easy to program for computer systems with IF, THEN, ELSE statements

The C4.5 algorithm has many features such as it can dealing with both continuous and discrete attributes, missing values and pruning trees after construction and it produce more accurate model.

3.4 Network Assumption

In this project, Azab Company is taken as case-study to building the proposed hybrid intrusion detection system. To detect the some types of flooding attacks that commonly populated and it threats most organization.

Azab is the one of the biggest company in sudan that attempt to do business investment in telecommunication sectors**,** construction and industrial projects. The security mechanism that was used in this company was became inefficient to detect the popular types of flooding based or distributed denial of service attacks and it's need to more effective security mechanism to detect the malicious activities that will threats the most types of companies and cause denial of services.

Chapter IV

Implementation And Results Analysis

4.1 Introduction

This chapter contain implementation for the proposed hybrid intrusion detection system and analysis the results. The implementation of the proposed system requires the definition and installing of many packets.

4.2 Installation Steps

The preparation of the proposed detection system must be done firstly by installing all the prerequisites packages from the ubuntu repositories:

```
$ sudo apt-get install –y openssl libpcre3-dev libssl-dev build-essential libpcap-dev libdumbnet-dev bison flex zlib1g-dev liblzma-dev
```

The preparation of intrusion detection environment was done through different steps the divided to many points to make the system will able to detect the attacks

4.2.1 Installing snort packages

These important packages that must be installed:

- openssl and libssl-dev: Provides security feature.
- build-essential**:** it need for compiling software.
- bison, flex: it used as parser.
- libpcre3-dev: Library of functions to support regular expressions required by snort.
- libpcap-dev: Library for traffic capture .

After that will create snort-src folder in home directory to download, extract, configure and install the packages such as snort and Data Acquisition library (DAQ) .

After the download of the source (e.g.snort) then will need to extract it and navigated to it's by using this command :

```
$cd snort-2.9.7.6                    # To navigate to it's to install and configure iit.
$./configure --enable-sourcefire    # To    configure/setting up snort
$make
$sudo make install                           # To install snort package
```

Through previous steps the installation of snort was done and the of snort -v command will show the version of installed snort.After that will need to configure snort to work as network intrusion detection.

4.2.2 Configuration of snort to run as network intrusion detection system:

Firstly , we need to create new user and group with the name snort to owener them some files.

```
$sudo useradd snort -r -s /sbin/nologin -c SNORT_IDS -g snort   # To create new snort user.
$sudo groupadd snort       # To create new snort group
```

Next, will need to create a number of files and folders that snort expects when running in NIDS mode. It was building as the following structure:

```
/etc/snort/rules/local.rules     # to assign your own rules
/etc/snort/sid-msg.map         # to define the snort’s signature ID.
```

Then must change the ownership of those files to our new snort user. Snort stores the configuration files in /etc/snort.conf and it stores the rules in etc/snort/rule and stores its logs in /var/log/snort**.**

Secondly, the definition of the network address that monitor it must done on this step vi modification of snort configuration file (/etc/snort/snort.conf) in the line to ipvar HOME_NET 172.16.34.0/24 and EXTERNAL_NET to any.

Finally, we want to enable one included rule file: /etc/snort/rules/local.rules by uncommitted it to enable the definition of our rules. And test the configuration of snort by running this command:

```
sudo snort -T -c /etc/snort/snort.conf -i ens32
   # option T for test , c for specify the conf file and i for determination of NIC
```

The output of this command is :

```
Snort successfully validated the configuration!
Snort exiting
```

### It’s mean that the snort was configured successfully and can add and use the rules that is need for the detection.

4.3 Building of signature detection model

The signature detection method will need to define signatures or rules to detect the types of attacks that match the predefined signatures then the next phase is to define the signatures and test it as in the next following points.

4.3.1. Writing and Testing Snort’s Rule

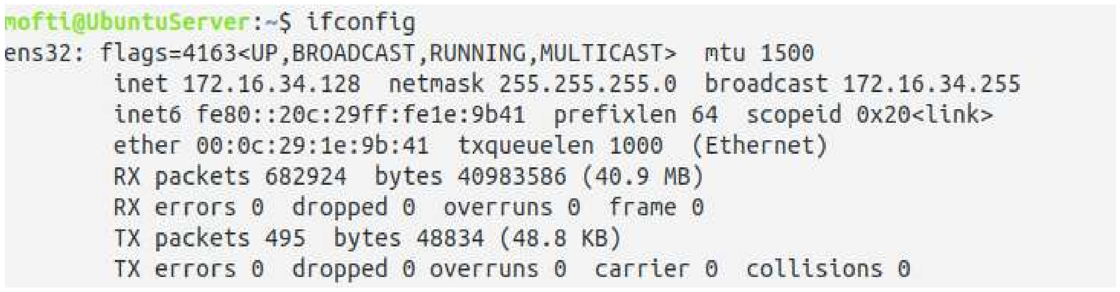
The intrusion detection system must full-filled with known signature if it’s based on signature detection technique..In the project will define signatures for flooding attacks like ICMP flooding and make snort to generate alert when the signatures are matches.

The definition of rules will done by edit the intended rules in this file /etc/snort/rules/local.rules such as the definition of ICMP messages:

alert icmp any any -> $HOME_NET any (msg:"ICMP test detected"; GID:1; sid:10000001; rev:001; classtype:icmp-event;)

This rule says is that for any ICMP packets it sees from any network to our home network, generate an alert with the message “ ICMP test. detected” The other information here (GID, REV, classtype) are used for group the rule, then will need to make sure this path /etc/snort/rules/local.rules that found in snort.conf file is uncommitted.

Before testing the rules the most important thing that must known is the internet protocol address of target networks or devices and this is will done through typing ifconfig command on linux operating system as seen in figure 4.1:



```
mofti@UbuntuServer:~$ ifconfig
ens32: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.16.34.128  netmask 255.255.255.0  broadcast 172.16.34.255
        inet6 fe80::20c:29ff:fe1e:9b41  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:1e:9b:41  txqueuelen 1000  (Ethernet)
        RX packets 682924  bytes 40983586 (40.9 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 495  bytes 48834 (48.8 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figure 4.1 : Information about network interface

After that will need to make snort to works as NIDS to monitor the traffic vi running this command:

$sudo snort -A console -q -u snort -g snort -c /etc/snort/snort.conf -i ens32

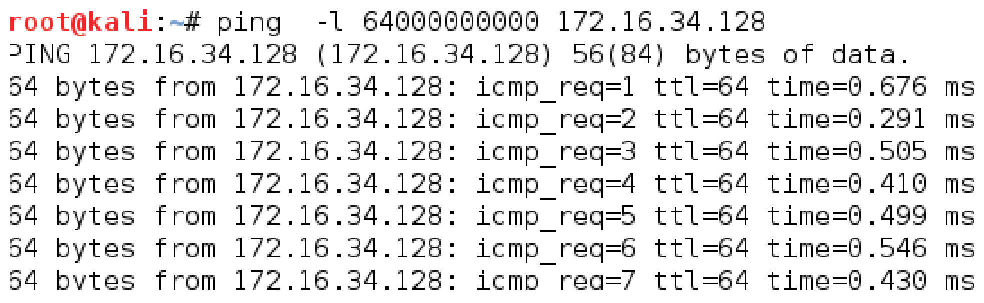
Note:

-A console ~ it print alerts in terminal interface

-q : Quiet. Don't show extra details ,-u snort : run snort as snort user , -g snort :run snort as snort group, -c /etc/snort/snort.conf ~ the path of snort configuration file

-i ens32~ the interface to listen on

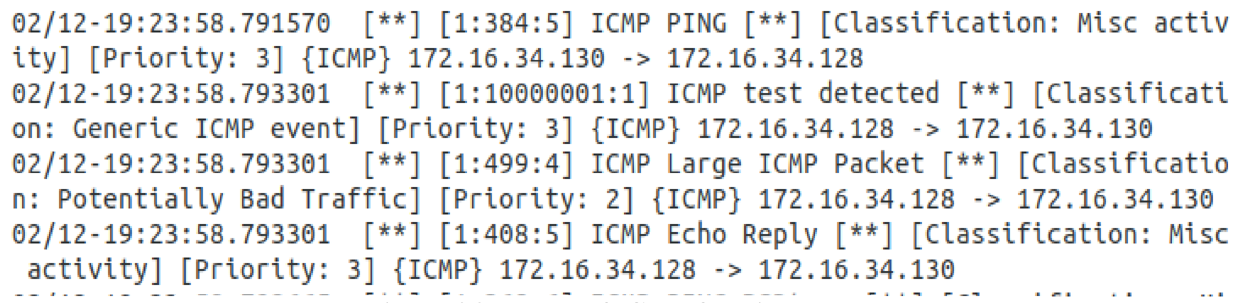
The test of snort will done from other terminal or device will pinging the Target destination (ip address: 172.16.34.128) with 64000000 packets size as will show in th figure 4.2:



```
root@kali:~# ping  -l 64000000000 172.16.34.128
PING 172.16.34.128 (172.16.34.128) 56(84) bytes of data.
64 bytes from 172.16.34.128: icmp_req=1 ttl=64 time=0.676 ms
64 bytes from 172.16.34.128: icmp_req=2 ttl=64 time=0.291 ms
64 bytes from 172.16.34.128: icmp_req=3 ttl=64 time=0.505 ms
64 bytes from 172.16.34.128: icmp_req=4 ttl=64 time=0.410 ms
64 bytes from 172.16.34.128: icmp_req=5 ttl=64 time=0.499 ms
64 bytes from 172.16.34.128: icmp_req=6 ttl=64 time=0.546 ms
64 bytes from 172.16.34.128: icmp_req=7 ttl=64 time=0.430 ms
```

Figure 4.2 : Pinging the target

The monitoring events for the behavior of flooding ICMP messages will show it if will return to the terminal of IDS system as it see in figure 4.3:



```
02/12-19:23:58.791570  [**] [1:384:5] ICMP PING [**] [Classification: Misc activ
ity] [Priority: 3] {ICMP} 172.16.34.130 -> 172.16.34.128
02/12-19:23:58.793301  [**] [1:10000001:1] ICMP test detected [**] [Classificati
on: Generic ICMP event] [Priority: 3] {ICMP} 172.16.34.128 -> 172.16.34.130
02/12-19:23:58.793301  [**] [1:499:4] ICMP Large ICMP Packet [**] [Classificatio
n: Potentially Bad Traffic] [Priority: 2] {ICMP} 172.16.34.128 -> 172.16.34.130
02/12-19:23:58.793301  [**] [1:408:5] ICMP Echo Reply [**] [Classification: Misc
 activity] [Priority: 3] {ICMP} 172.16.34.128 -> 172.16.34.130
```

Figure 4.3: Detect of ICMP flooding attacks

The resulting of ICMP flooding attacks cause the high utilization of resources (CPU , memory , buffer of network) as will represented in figure 4.4 :

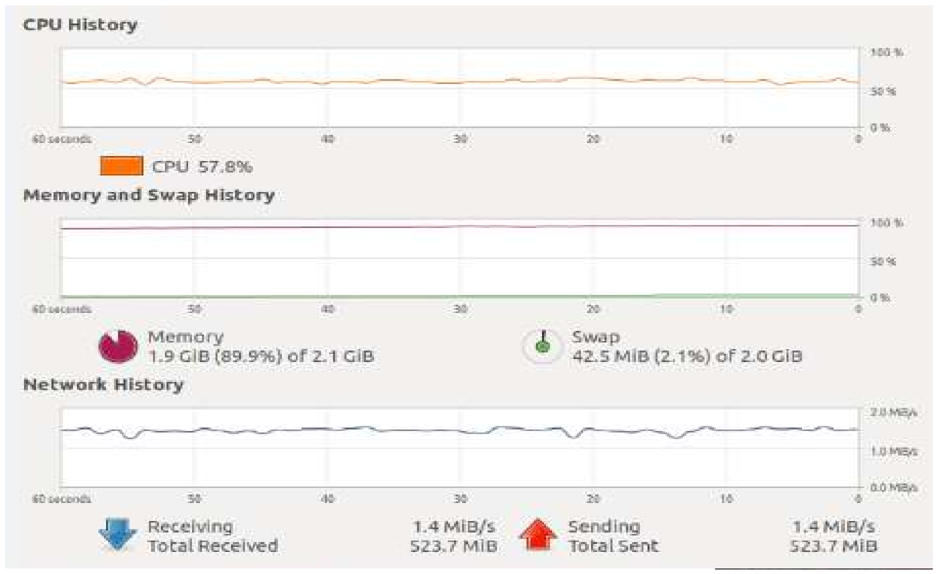


Figure 4.4: effect of ICMP flooding messages

4.3.2. Load balancing on snort server

The snort server always work in detection mode to monitor all the traffic and this cause high load to it's, so that to reduce the load will need to install and configure barnyard2.

First, will need to install some pre-requisites packages:

```
$sudo apt-get install -y mysql-server libmysqlclient-dev mysql-client autoconf libtool
```

The abc123# will choose it as root password.

Next, will need to edit the snort.conf file to tell snort to output event in a binary form through using unified2 binary format by adding this line :

```
output unified2: filename snort.u2, limit 128
```

Secondly, will come to installation and configuration of Barnyard2:

Before the installation, will need to download the source of packet through using following command:

```
wget https:// github.com/firnsy/barnyard2/archive/7254c24702392288fe6be9
48f88afb74040f6dc9.tar.gz -O barnyard2-2-1.14-336.tar.gz.
```

Then will extract the source and move the files to barnyard2-2-1.14-336 to rightly configure it.

After that, the determination of MYSQL library packet that needed to compatible it with Barnyard2 by running this command :

```
$./configure --with-mysql --with-mysql-libraries=/usr/lib/x86_64-linux-gnu
```

Then continue with the install:

```
$ make
$sudo make install    # to install barnyard2
```

Barnyard2 is now installed to /usr/local/bin/barnyard2. To configure snort to use Barnyard2, we need to copy a few files from the source package:

```
$cd ~/snort_src/barnyard2-2-1.14-336
$sudo cp etc/barnyard2.conf /etc/snort    # this step for coping
```

Then will need to create barnyard2 diracrty in log directory and change the ownership to snort user and group

```
$sudo mkdir    /var/log/barnyard2
 $sudo touch /var/log/snort/barnyard2.mofti
Sudo chown snort.snort /var/log/snort/barnyard2.mofti
$sudo touch /etc/snort/sid-msg.map
```

Thirdly, the create of the database must be done to make Barnyard2 save alert to the database by setting and typing this commands :

$ mysql -u root -p

mysql> create database snort;   # To create snort DB

mysql> use snort;   # To use snort DB

mysql> source ~/snort_src/barnyard2-2-1.13/schemas/create_mysql

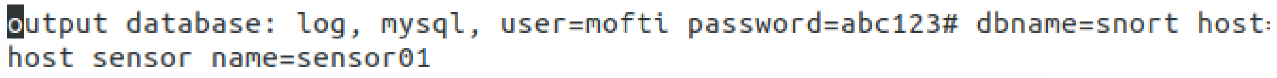
mysql> CREATE USER 'mofti'@'localhost' IDENTIFIED BY 'abc123#';

mysql> grant create, insert, select, delete, update on snort.* to 'moftit'@'localhost';

# To give mofti authority for dealing with DB .

mysql> exit

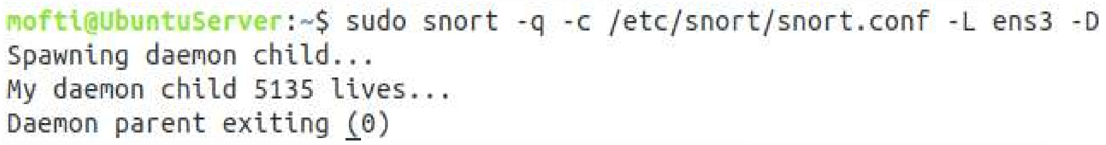
Now that the snort database has been created, we need to tell Barnyard2 about the details of the database so that will need to edit to the Barnyard2 configuration file: $sudo vi /etc/snort/barnyard2.conf and at the end of the file, append this line that it show in figure 4.5:



```
output database: log, mysql, user=mofti password=abc123# dbname=snort host=
host sensor name=sensor01
```

Figure 4.5: Information about the db connection

After the configuration of Barnyard2 to work with snort, will need to test them together and generate alert that require to run snort as a daemon though using this command to run snort as process or daemon as will see it in figure 4.6:



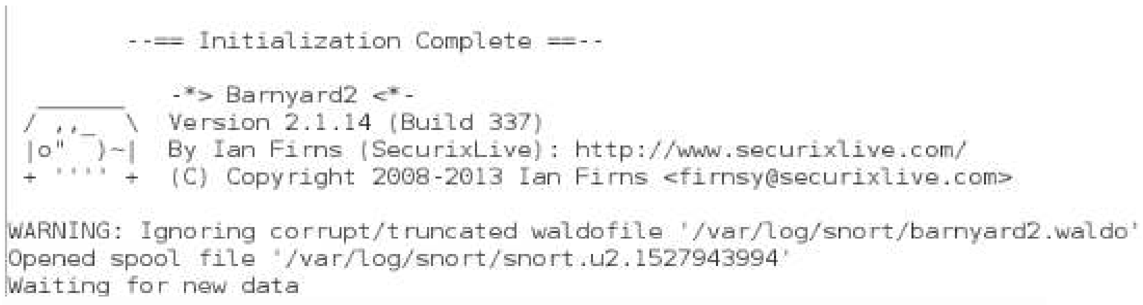
```
mofti@UbuntuServer:~$ sudo snort -q -c /etc/snort/snort.conf -L ens3 -D
Spawning daemon child...
My daemon child 5135 lives...
Daemon parent exiting (0)
```

Figure 4.6: Creation of the daemon

Then Pinging the IP address (172.16.34.128) of the interface specified above ens32). If you check snort’s log directory, you should see a file called snort.u2.12356**.** These are the binary alerts that snort has written out for Barnyard2 to process.

Now we want to tell Barnyard2 to look at these events and load into the snort database instance. We run Barnyard2 with the following flags:

sudo barnyard2 –c /etc/snort/barnyard2.conf -d /var/log/snort -f snort.u2 -w /var/log/snort/barnyard2.waldo -g snort -u snort and It give this output as it see in figure 4.7 :



```
        --== Initialization Complete ==--

  ______     -*> Barnyard2 <*-
 / ,,_  \   Version 2.1.14 (Build 337)
 |o"  )~|   By Ian Firns (SecurixLive): http://www.securixlive.com/
 + '''' +   (C) Copyright 2008-2013 Ian Firns <firnsy@securixlive.com>

WARNING: Ignoring corrupt/truncated waldofile '/var/log/snort/barnyard2.waldo'
Opened spool file '/var/log/snort/snort.u2.1527943994'
Waiting for new data
```

Figure 4.7: Information about barnyard

The stopping of barnyard2 from running will done by using ctrl+c , then to stop the snort daemon must use ps to find and terminate it by typing sudo kill <pid>.

4.3.3. Installing and configuring BASE

BASE is graphical web interface that is used to mointer network activities and provide efficient for most public that deals with graphical user interface and it's difficult for doing administrative task by using the command line so that will need to customize the GUI web application like BASE interface to help for solving the problems. BASE is a simple web GUI for snort, it's easy for configuration and simple to use .

There are some pre-requested packets that must be installed such as:

1. sudo add-apt-repository ppa:ondrej/php
2. sudo apt-get install -y apache2 libapache2-mod-php5.6 php5.6-mysql php5.6-cli php5.6 php5.6-common php5.6-gd php5.6-cli php-pear php5.6-xml
3. sudo pear install -f --alldeps Image_Graph # To install Pear image graph.Then will need to download and install ADODB**.**

ADODB is a PHP database class library to provide more powerful abstractions for performing queries and managing databases

cd ~/snort_src

$wget https:// sourceforge.net/projects/adodb/files/adodb-php5-only/adodb-520-for-php5/adodb-5.20.8.tar.gz   # To download the packet

then will need to extract archive folder and navigate to adodb source and change the ownership of it's by using this command :

#sudo chmod -R 755 /var/adodb   # To set specific permission

Next step is to download BASE and copy to apache root through running these commands:

```
$cd ~/snort_src
$wgethttp://sourceforge.net/projects/secureideas/files/BASE/base- .4.5/base-1.4.5.tar.gz
$tar xzvf base-1.4.5.tar.gz
$sudo mv base-1.4.5 /var/www/html/base/
```

After that will need to create the BASE configuration file by using this command:

```
$cd /var/www/html/base # to nabigate to base folder
$ sudo cp base_conf.php.dist base_conf.php
```

Now edit the config file:

```
$sudo vi /var/www/html/base/base_conf.php
```

With the following settings (note that the trailing slash on line 80 is required, despite the instructions in the configuration file):

```
$BASE_urlpath = '/base';              # line 50
$DBlib_path = '/var/adodb/';          #line 80
$alert_dbname     = 'snort';          # line 102
$alert_host       = 'localhost';
$alert_user       = 'snort';
$alert_password   = 'MySqlSNORTpassword';   # line 106
```

Then restart Apache to update the setting and configuration:

```
$sudo service apache2 restart       # Then restart the apache service
```

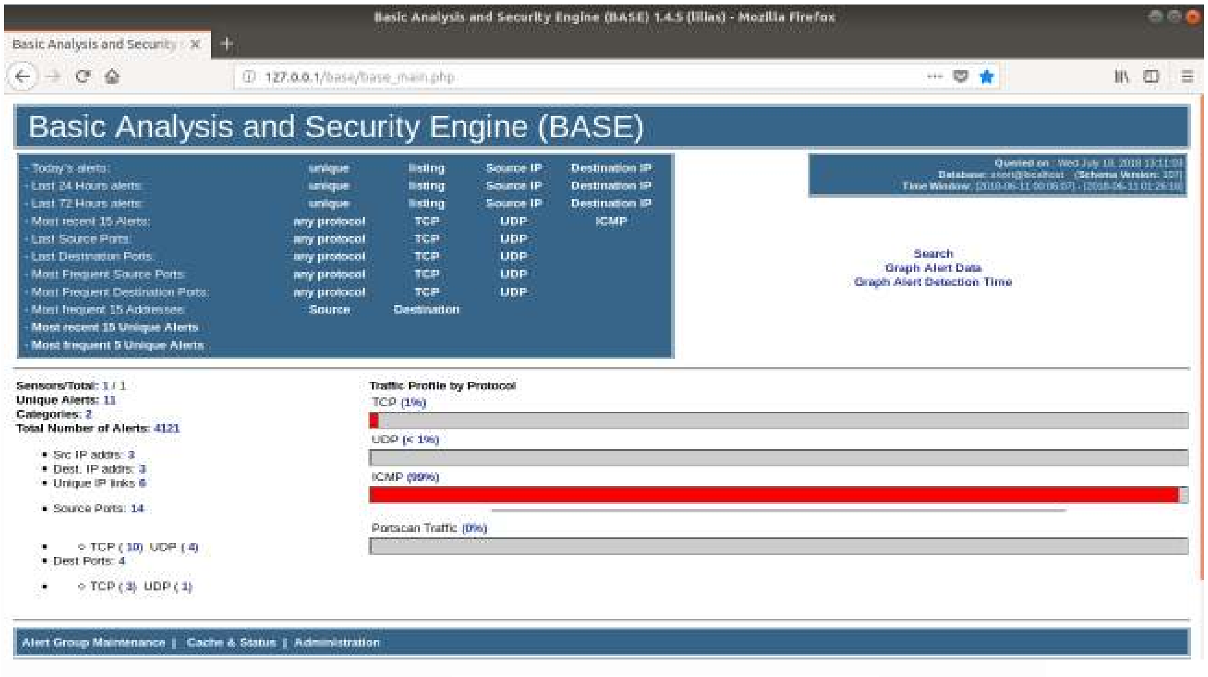
Then will generate ICMP flooding messges to the target device by running this command:

```
    ping 172.16.34.128 –s 65500       # s for determine the size of messages
```

The result of running that command is sending flooding message with the 655000 bytes

The last step to configure BASE is done via http:

1. Browse to http://localhost/base/index.php and click on the setup page link.
2. Click on the Create BASE AG button on the upper right of the page.
3. Click on the main page link to show the types of attacks that was occurs and it display the effect ion of flooding as in figure 4.8:


Basic Analysis and Security Engine (BASE) 1.4.5 (lilias) - Mozilla Firefox

127.0.0.1/base/base_main.php

Basic Analysis and Security Engine (BASE)

- Today's alerts:	unique	listing	Source IP	Destination IP
- Last 24 Hours alerts:	unique	listing	Source IP	Destination IP
- Last 72 Hours alerts:	unique	listing	Source IP	Destination IP
- Most recent 15 Alerts:	any protocol	TCP	UDP	ICMP
- Last Source Ports:	any protocol	TCP	UDP	
- Last Destination Ports:	any protocol	TCP	UDP	
- Most Frequent Source Ports:	any protocol	TCP	UDP	
- Most Frequent Destination Ports:	any protocol	TCP	UDP	
- Most frequent 15 Addresses:	Source	Destination		
- Most recent 15 Unique Alerts				
- Most frequent 5 Unique Alerts				

Queried on : Wed July 18, 2018 13:11:03
Database: snort@localhost (Schema Version: 107)
Time Window: [2018-06-11 00:06:07] - [2018-06-11 01:26:16]

Search
Graph Alert Data
Graph Alert Detection Time

Sensors/Total: 1 / 1
Unique Alerts: 11
Categories: 2
Total Number of Alerts: 4121

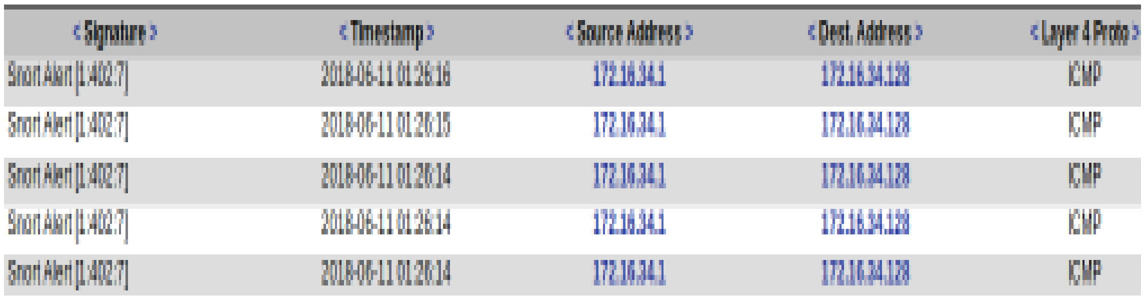
- Src IP addrs: 3
- Dest. IP addrs: 3
- Unique IP links 6
- Source Ports: 14
- ◦ TCP (10) UDP (4)
- Dest Ports: 4
- ◦ TCP (3) UDP (1)

Traffic Profile by Protocol
TCP (1%)
UDP (< 1%)
ICMP (99%)
Portscan Traffic (0%)

Alert Group Maintenance | Cache & Status | Administration

Figure 4.8: Interface of base module

From the previous figure 4.8 the red line illustrate that is ICMP flooding message , to show the details of sources and destination of the messages click on ICMP tag and it's display in figure 4.9 :



< Signature >	< Timestamp >	< Source Address >	< Dest. Address >	< Layer 4 Proto >
Snort Alert [1:402:7]	2018-06-11 01:26:16	172.16.34.1	172.16.34.128	ICMP
Snort Alert [1:402:7]	2018-06-11 01:26:15	172.16.34.1	172.16.34.128	ICMP
Snort Alert [1:402:7]	2018-06-11 01:26:14	172.16.34.1	172.16.34.128	ICMP
Snort Alert [1:402:7]	2018-06-11 01:26:14	172.16.34.1	172.16.34.128	ICMP
Snort Alert [1:402:7]	2018-06-11 01:26:14	172.16.34.1	172.16.34.128	ICMP

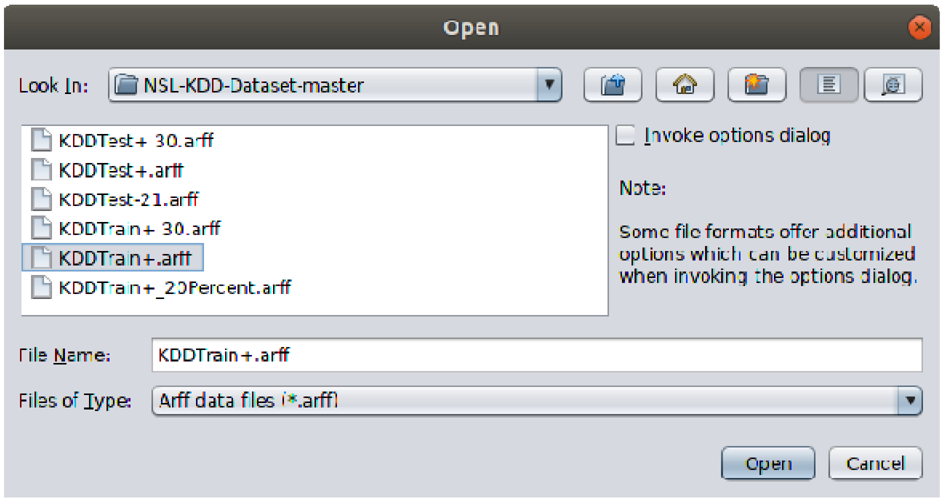
Figure 4.9: Information about the interfaces and protocol

4.4 Building of anomaly detection model

Anomaly detection model need a profile or model to classify a traffics the model will create it by using data mining techniques (Decision Tree and Neural Network) under using weka tools through the following parts after opening weka by using this command : java -jar weka.jar then will need to importing data-set.

4.4.1 Importing and preprocessing data-set:

The building of anomaly detection model require processed imported data set before training it by using the specified algorithm ,so that that will need to import the NSL -KDD train data -set as represented by figure 4.10:


Open

Look In: NSL-KDD-Dataset-master

KDDTest+ 30.arff
KDDTest+.arff
KDDTest-21.arff
KDDTrain+ 30.arff
KDDTrain+.arff
KDDTrain+_20Percent.arff

Invoke options dialog

Note:

Some file formats offer additional options which can be customized when invoking the options dialog.

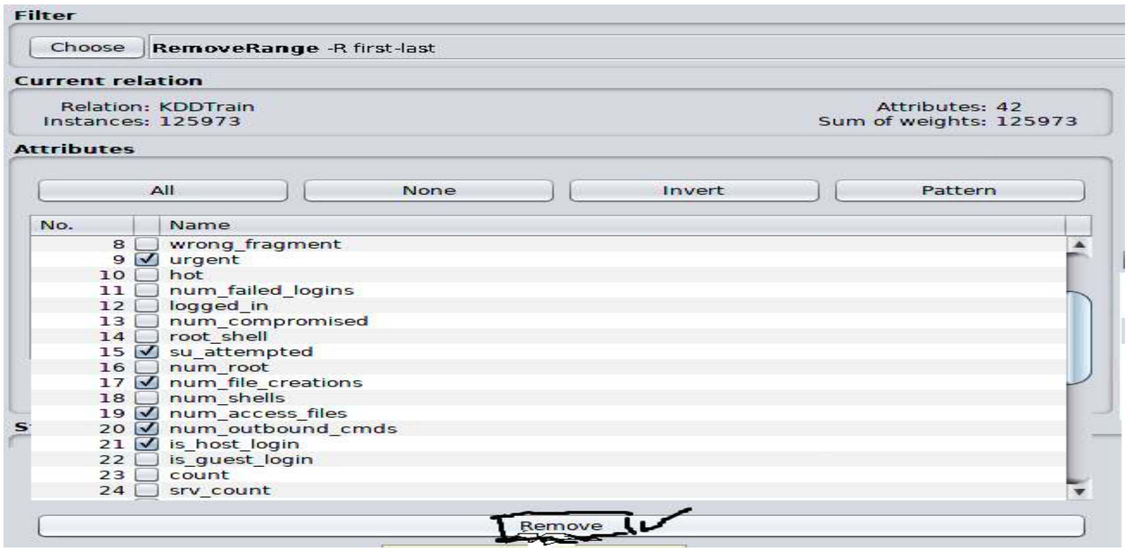
File Name: KDDTrain+.arff

Files of Type: Arff data files (*.arff)

Open Cancel

Figure 4.10: Importing of data-set

After that, will reprocess it by extracting the important features and cleaning it from any noisy or duplicated values all that done on total 42 features as it appear in figure 4.11:


Filter

Choose RemoveRange -R first-last

Current relation

Relation: KDDTrain
Instances: 125973
Attributes: 42
Sum of weights: 125973

Attributes

All None Invert Pattern

No.		Name
8	☐	wrong_fragment
9	✓	urgent
10	☐	hot
11	☐	num_failed_logins
12	☐	logged_in
13	☐	num_compromised
14	☐	root_shell
15	✓	su_attempted
16	☐	num_root
17	✓	num_file_creations
18	☐	num_shells
19	✓	num_access_files
20	✓	num_outbound_cmds
21	✓	is_host_login
22	☐	is_guest_login
23	☐	count
24	☐	srv_count

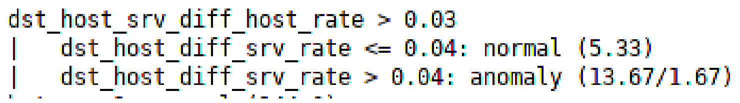
Remove

Figure 4.11: Removing irreverent features

42 feature was processed it and extracted to become 30 features that only need for the detection to pass it to algorithms.

4.4.2. Working of data mining algorithms

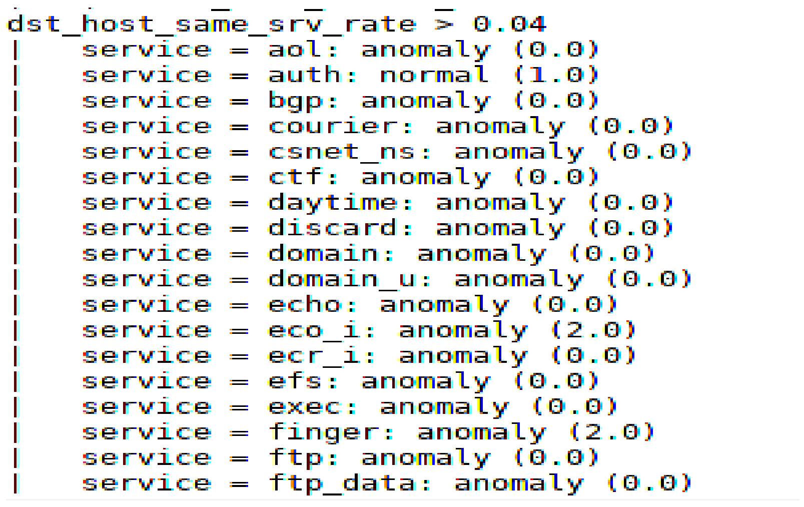
After processing data set then will pass it to enhanced decision tree algorithm to classify a traffic to normal and abnormal, such as the following figures 4.12:



```
dst_host_srv_diff_host_rate > 0.03
|   dst_host_diff_srv_rate <= 0.04: normal (5.33)
|   dst_host_diff_srv_rate > 0.04: anomaly (13.67/1.67)
```

Figure 4.12: Classification of traffics base on no of packets

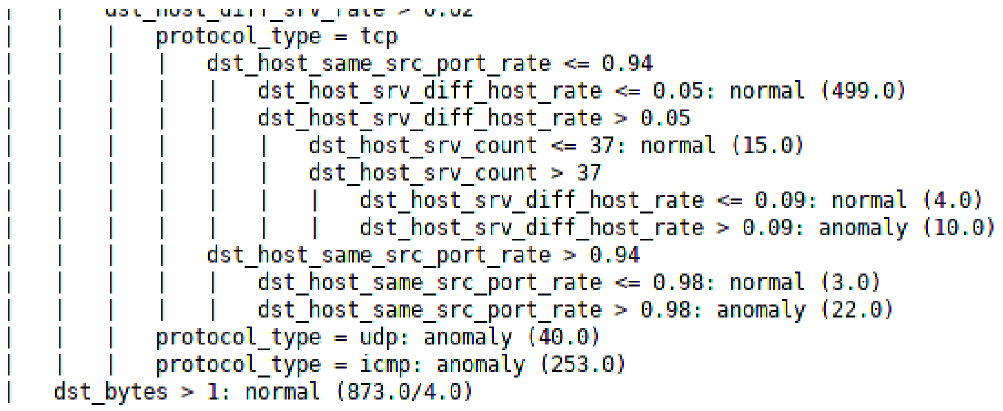
The previous figure differentiate the behavior of valid users form invalid users Attacker) according to number of packet that are sent. If there are .04 (4) packets or less than it as inbound traffic this mean the normal traffic for valid user else this is abnormal traffics that is sent by attacker that act as ICMP flooding attacks or may be use other type of flooding attacks as it see in figure 4.13 :



```
dst_host_same_srv_rate > 0.04
|   service = aol: anomaly (0.0)
|   service = auth: normal (1.0)
|   service = bgp: anomaly (0.0)
|   service = courier: anomaly (0.0)
|   service = csnet_ns: anomaly (0.0)
|   service = ctf: anomaly (0.0)
|   service = daytime: anomaly (0.0)
|   service = discard: anomaly (0.0)
|   service = domain: anomaly (0.0)
|   service = domain_u: anomaly (0.0)
|   service = echo: anomaly (0.0)
|   service = eco_i: anomaly (2.0)
|   service = ecr_i: anomaly (0.0)
|   service = efs: anomaly (0.0)
|   service = exec: anomaly (0.0)
|   service = finger: anomaly (2.0)
|   service = ftp: anomaly (0.0)
|   service = ftp_data: anomaly (0.0)
```

Figure 4.13: Classification of traffics

Attackers can also cause denial of services to services that provides to user by using TCP or UDP flooding messages to full the destination's buffers with large a mounts of traffics that represented in the figure 4.14 :



```
|   |   dst_host_diff_srv_rate > 0.02
|   |   |   protocol_type = tcp
|   |   |   |   dst_host_same_src_port_rate <= 0.94
|   |   |   |   |   dst_host_srv_diff_host_rate <= 0.05: normal (499.0)
|   |   |   |   |   dst_host_srv_diff_host_rate > 0.05
|   |   |   |   |   |   dst_host_srv_count <= 37: normal (15.0)
|   |   |   |   |   |   dst_host_srv_count > 37
|   |   |   |   |   |   |   dst_host_srv_diff_host_rate <= 0.09: normal (4.0)
|   |   |   |   |   |   |   dst_host_srv_diff_host_rate > 0.09: anomaly (10.0)
|   |   |   |   dst_host_same_src_port_rate > 0.94
|   |   |   |   |   dst_host_same_src_port_rate <= 0.98: normal (3.0)
|   |   |   |   |   dst_host_same_src_port_rate > 0.98: anomaly (22.0)
|   |   |   protocol_type = udp: anomaly (40.0)
|   |   |   protocol_type = icmp: anomaly (253.0)
|   dst_bytes > 1: normal (873.0/4.0)
```

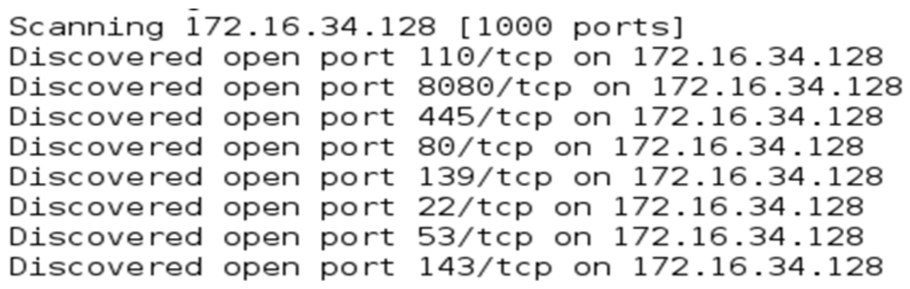
Figure 4.14: Classification of TCP traffics

After the building the proposed enhanced hybrid instruction detection with hybrid approach signature and anomaly based detection that it based on the decision tree then will test the system with many types of attacks behaviors.

4.5 Testing of attacks behaviors

4.5.1 Nmap scanning attacks

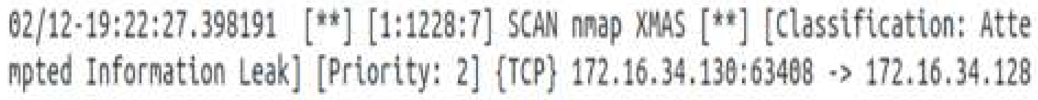
This type of attacks attempt to obtain information a bout the target device such as the information of the used of operating system and open and closed ports as in figure 4.15 after using the nmap –A –v 172.16.34.128 command :



```
Scanning 172.16.34.128 [1000 ports]
Discovered open port 110/tcp on 172.16.34.128
Discovered open port 8080/tcp on 172.16.34.128
Discovered open port 445/tcp on 172.16.34.128
Discovered open port 80/tcp on 172.16.34.128
Discovered open port 139/tcp on 172.16.34.128
Discovered open port 22/tcp on 172.16.34.128
Discovered open port 53/tcp on 172.16.34.128
Discovered open port 143/tcp on 172.16.34.128
```

Figure 4.15: Output of scanning process

Figure 4.15 illustrate that there are many open tcp ports such as 445,80 and 53.When the traffics of scanning process come to the instruction detection system then it compare the observed traffics with the signature in IDS system as in figure 4.16 if it’s matched the rules:



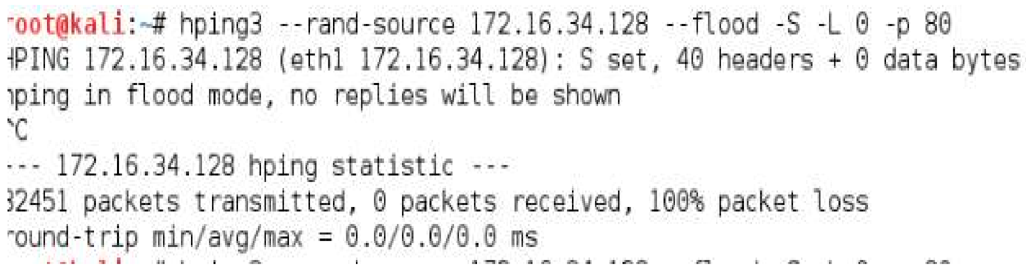
```
02/12-19:22:27.398191  [**] [1:1228:7] SCAN nmap XMAS [**] [Classification: Atte
mpted Information Leak] [Priority: 2] {TCP} 172.16.34.130:63408 -> 172.16.34.128
```

Figure 4.16: Result of detection nmap scanning attacks

Figure 4.16 show that there are malicious activities from nmap scanning attacks to attempted information leak and can use that information that it saw in figure 4.15 to attacks the target.

4.5.2 SYS flooding attacks

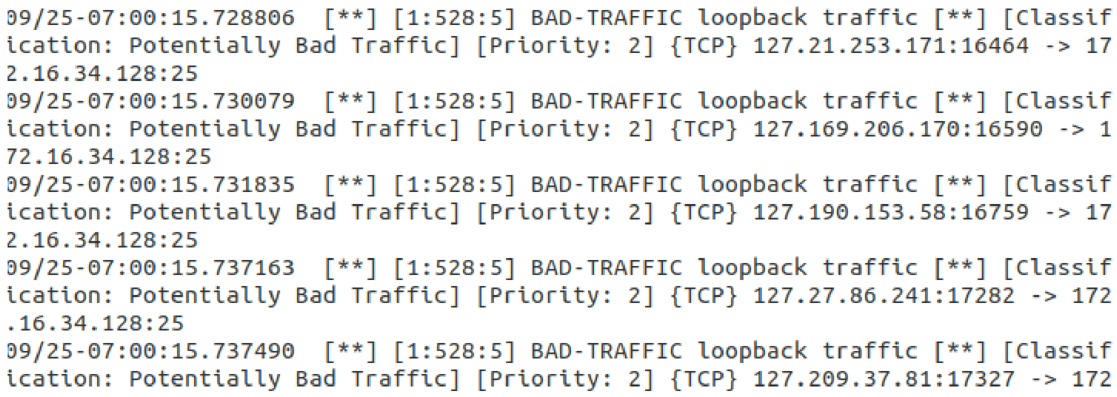
The behavior of this type of attacks was done by sending many SYN messages to the target and fill it’s buffers vi using this command as it found in figure 4.17:



```
'oot@kali:~# hping3 --rand-source 172.16.34.128 --flood -S -L 0 -p 80
HPING 172.16.34.128 (eth1 172.16.34.128): S set, 40 headers + 0 data bytes
ping in flood mode, no replies will be shown
^C
--- 172.16.34.128 hping statistic ---
32451 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

Figure 4.17: Command for SYS flooding attacks

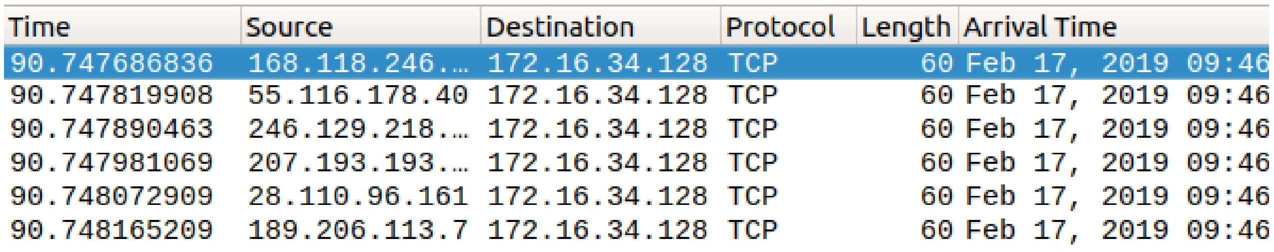
The target (HIDS) system cause alerts message after it receive abnormal traffics of sys flooding attacks and cause the flowing results of detection as in figure 4.18:



```
09/25-07:00:15.728806  [**] [1:528:5] BAD-TRAFFIC loopback traffic [**] [Classif
ication: Potentially Bad Traffic] [Priority: 2] {TCP} 127.21.253.171:16464 -> 17
2.16.34.128:25
09/25-07:00:15.730079  [**] [1:528:5] BAD-TRAFFIC loopback traffic [**] [Classif
ication: Potentially Bad Traffic] [Priority: 2] {TCP} 127.169.206.170:16590 -> 1
72.16.34.128:25
09/25-07:00:15.731835  [**] [1:528:5] BAD-TRAFFIC loopback traffic [**] [Classif
ication: Potentially Bad Traffic] [Priority: 2] {TCP} 127.190.153.58:16759 -> 17
2.16.34.128:25
09/25-07:00:15.737163  [**] [1:528:5] BAD-TRAFFIC loopback traffic [**] [Classif
ication: Potentially Bad Traffic] [Priority: 2] {TCP} 127.27.86.241:17282 -> 172
.16.34.128:25
09/25-07:00:15.737490  [**] [1:528:5] BAD-TRAFFIC loopback traffic [**] [Classif
ication: Potentially Bad Traffic] [Priority: 2] {TCP} 127.209.37.81:17327 -> 172
```

Figure 4.18: Result of detection of TCP /SYS flooding attacks

Figures 4.18 notify that, there are bad traffics of tcp or sys messages with the flowing source ip address and destination ip address that are displayed in figure 4.19:



Time	Source	Destination	Protocol	Length	Arrival Time
90.747686836	168.118.246.…	172.16.34.128	TCP	60	Feb 17, 2019 09:46
90.747819908	55.116.178.40	172.16.34.128	TCP	60	Feb 17, 2019 09:46
90.747890463	246.129.218.…	172.16.34.128	TCP	60	Feb 17, 2019 09:46
90.747981069	207.193.193.…	172.16.34.128	TCP	60	Feb 17, 2019 09:46
90.748072909	28.110.96.161	172.16.34.128	TCP	60	Feb 17, 2019 09:46
90.748165209	189.206.113.7	172.16.34.128	TCP	60	Feb 17, 2019 09:46

Figure 4.19: Information about src and dst of messages

This flooding sys messages cause high load to the server that can be represented as in figure 4.20:

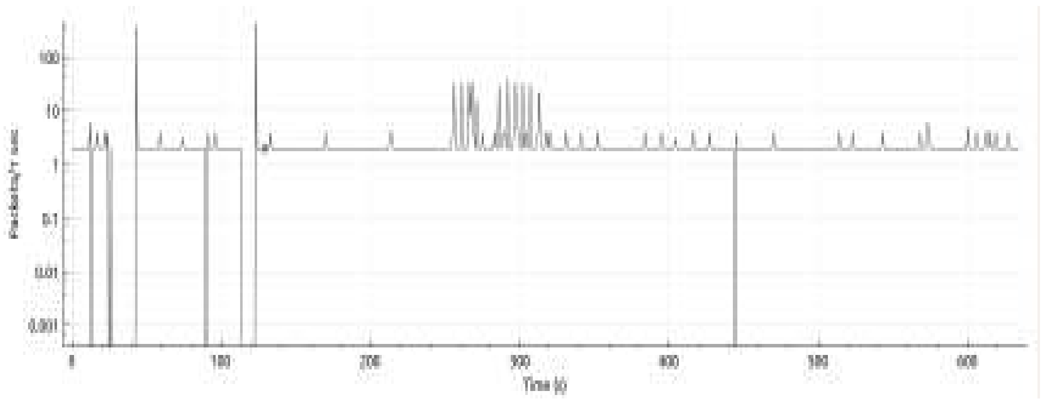


Figure 4.20: Effect of sys attacks on the resources

Figure 4.20 illustrates that the target devise it receive large amount of sys flooding messages from the hacker device and these amount of traffic to the target device make the result of increasing of the loads with the changing of the time that represented in x dimension in figure 4.20 .

4.5.3 UDP flooding attacks

This types of attacks are occurs by sending many udp packet to the target to slow the system by filling it storage capacity such as RAM and that effect to resources as it seen in figure 4.21:

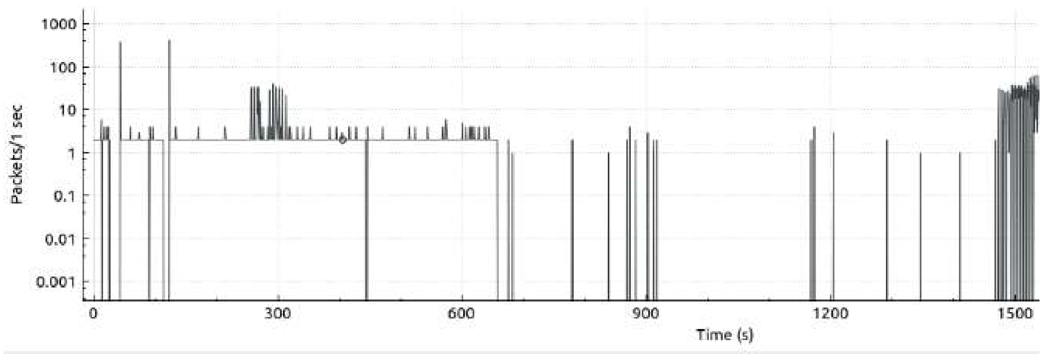
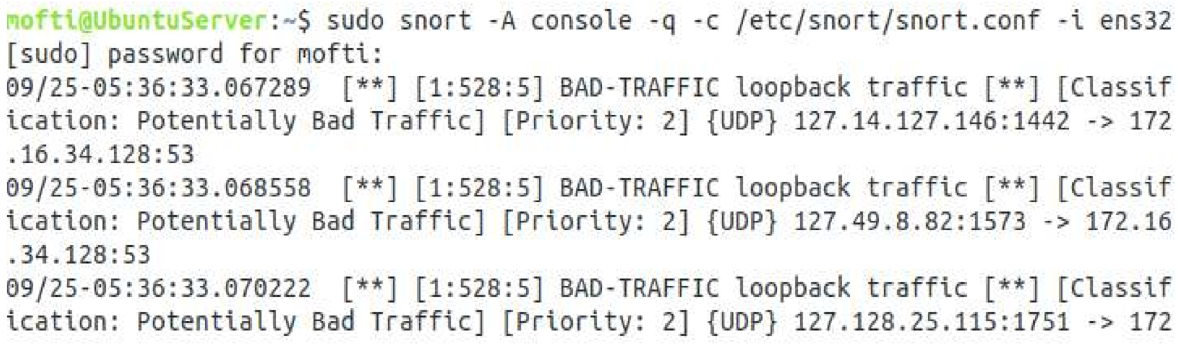


Figure 4.21: Effect of UDP flooding attacks on the resources

Figure 4.21 show that the number of packets is increasingly received and it cause denial to services.

Then the proposed hybrid IDS system is enable to detect these types of attacks by running this command to change to detection mode as in represented it in figure 4.22:

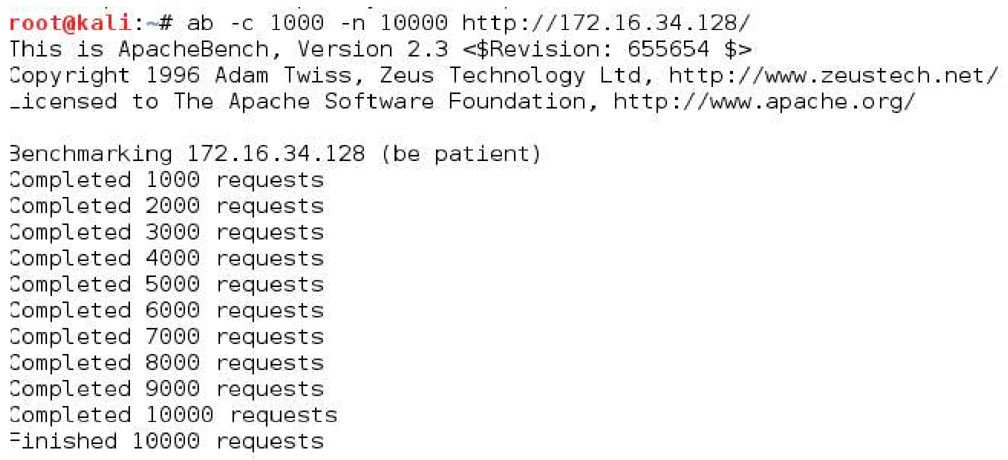


```
mofti@UbuntuServer:~$ sudo snort -A console -q -c /etc/snort/snort.conf -i ens32
[sudo] password for mofti:
09/25-05:36:33.067289  [**] [1:528:5] BAD-TRAFFIC loopback traffic [**] [Classif
ication: Potentially Bad Traffic] [Priority: 2] {UDP} 127.14.127.146:1442 -> 172
.16.34.128:53
09/25-05:36:33.068558  [**] [1:528:5] BAD-TRAFFIC loopback traffic [**] [Classif
ication: Potentially Bad Traffic] [Priority: 2] {UDP} 127.49.8.82:1573 -> 172.16
.34.128:53
09/25-05:36:33.070222  [**] [1:528:5] BAD-TRAFFIC loopback traffic [**] [Classif
ication: Potentially Bad Traffic] [Priority: 2] {UDP} 127.128.25.115:1751 -> 172
```

Figure 4.22: Result of detection UDP flooding attacks

4.5.4 HTTP Flooding attacks

This types of attacks was done by sending many requests to target to open channel or session that carry many inbound traffics this was occurs by running this command that it written in figure 4.23:



```
root@kali:~# ab -c 1000 -n 10000 http://172.16.34.128/
This is ApacheBench, Version 2.3 <$Revision: 655654 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 172.16.34.128 (be patient)
Completed 1000 requests
Completed 2000 requests
Completed 3000 requests
Completed 4000 requests
Completed 5000 requests
Completed 6000 requests
Completed 7000 requests
Completed 8000 requests
Completed 9000 requests
Completed 10000 requests
Finished 10000 requests
```

Figure 4.23: Result of sending many no of the requests

Figure 4.23 illustrate the number of request that send to the target that has ip address: 172.16.34.128 and it cause high load as represented in figure 4.24:

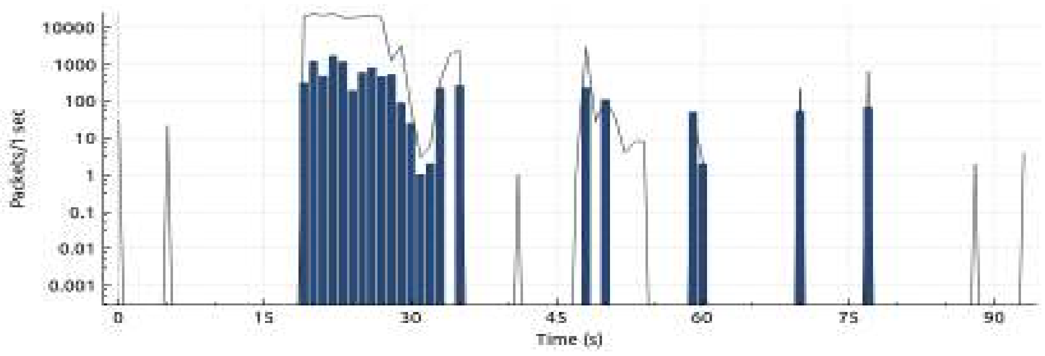
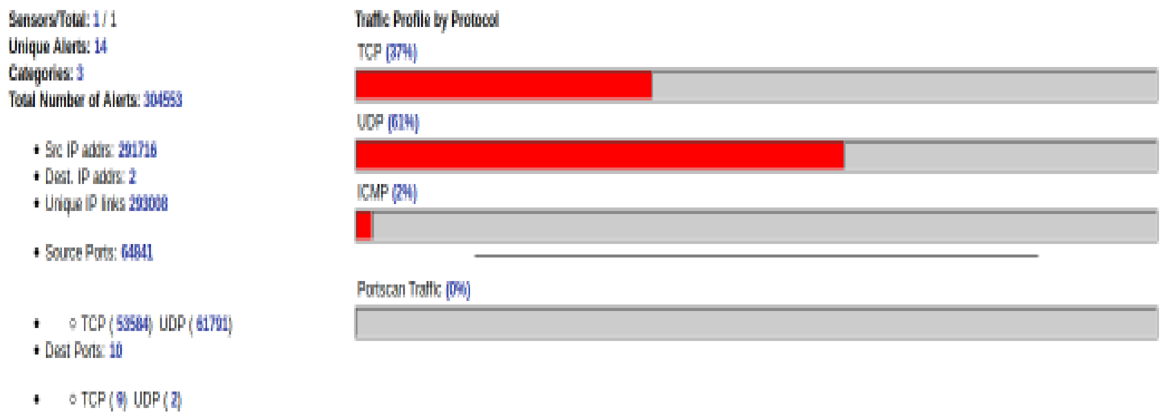


Figure 4.24: Load on the target device

Figure 4.24 show that the target device receive many inbound traffics that coming from unauthorized users and it cause high load with the large amount of packets that increasing with the changing of the time (s).

All these types of tested attacks will send flooding messages to the target to slow it performance and cause denial to provided services and it also effect to availability of it's by full-filled with the large amount of traffics and it was become difficult to process the received requested efficiency with good response time because there are a high traffic coming to the sever resulted from unauthorized activities to its as represented in the following figure 4.25:



```
Sensors/Total: 1 / 1
Unique Alerts: 14
Categories: 3
Total Number of Alerts: 304553

  • Src IP addrs: 291716
  • Dest. IP addrs: 2
  • Unique IP links 293008

  • Source Ports: 64841

  •     ○ TCP ( 53584) UDP ( 61791)
  • Dest Ports: 10

  •     ○ TCP ( 9) UDP ( 2)
```

```
Traffic Profile by Protocol
TCP (37%)
UDP (61%)
ICMP (2%)

Portscan Traffic (0%)
```

Figure 4.25: Amount of unauthorized traffics

The activities on the networks it need to continuously monitoring it to detect the unauthorized activities that cause flooding or unauthorized traffics to try to prevent it early before it cause high damage to the resources or effect to the provided services to the clients, so that there is high importance to the intrusion prevention system.

4.6 Intrusion Prevention System

Instruction prevention system is system that is configured to prevent or restrict the unauthorized activities or traffics and this is done by many system like firewall, pefsense , access control list and iptables .....etc.

4.6.1 Iptables:

Iptables is an application that allows users to configure specific rules that will be enforced by the kernel's netfilter framework. It acts as a packet filter and firewall that examines and directs traffic based on port, protocol and other criteria.

For example, the following command adds a rule to the beginning of the chain that will drop all packets from the address 172.16.34.132:

```
iptables -I INPUT -s 172.16.34.132 -j DROP
```

The samples:

-I: mean insert new rules

-s: Indicate for source address of the device.

-j: stands for jump. It specifies the target of the rule and what action will be performed if the packet is a match.

Some system administrators want to block incoming ping requests due to security concerns. While the threat is not that big and this is done by using this command:

```
# iptables -A INPUT -p icmp -i ens32 -j DROP
```

In case if only want to block TCP traffic from that IP address, i can use the -p option that specifies the protocol. That way the command will look like this:

```
# iptables -A INPUT -p tcp -s 172.16.34.32    -j DROP
```

You may use a port to block all traffic coming in on a specific interface. For example:

```
# iptables -A INPUT -j DROP -p tcp --destination-port 23 -i ens32
```

Description of the samples:

-A : To append new rules , -i : To specify the interface that implement the rules on it's , -m: is for match.

To allow incoming traffic from multiports must types this command:

```
# iptables -A INPUT -p tcp -m multiport --destination-ports
  2,25,53,80,443,465,5269,5280,8999:9003, 5222 -j ACCEPT
```

Iptables can use it to block all traffic and then only allow traffic from certain IP addresses. These firewall rules limit access to specific resources at the network layer. Below is an example sequence of commands:

```
iptables -A INPUT -s 172.16.0.0/16 -j ACCEPT
iptables -I INPUT -s 172.16.34.132 -j DROP
iptables -P INPUT DROP
iptables -P FORWARD DROP
```

The final two commands set the default policy for all inputs and forward chains to drop all packets. Then, to add new rules that disable the outgoing ICMP traffics must use this command:

```
# iptables -A OUTPUT -p icmp --icmp-type 8 -j DROP
```

The following iptables rule will block all incoming traffic from 172.16.34.128 where source port is port 80 / www and it's used to block an access to a specific website as in figure by using the following commamd:

```
# iptables -A INPUT -s 172.16.34.128    -p tcp --sport 80 -j DROP
```

Figure 4.26 illustrate that the status of the target before it flooded and the status of its after the flooding it with the large amount of inbound traffics that represented with high curve as it see in figure 4.26 :

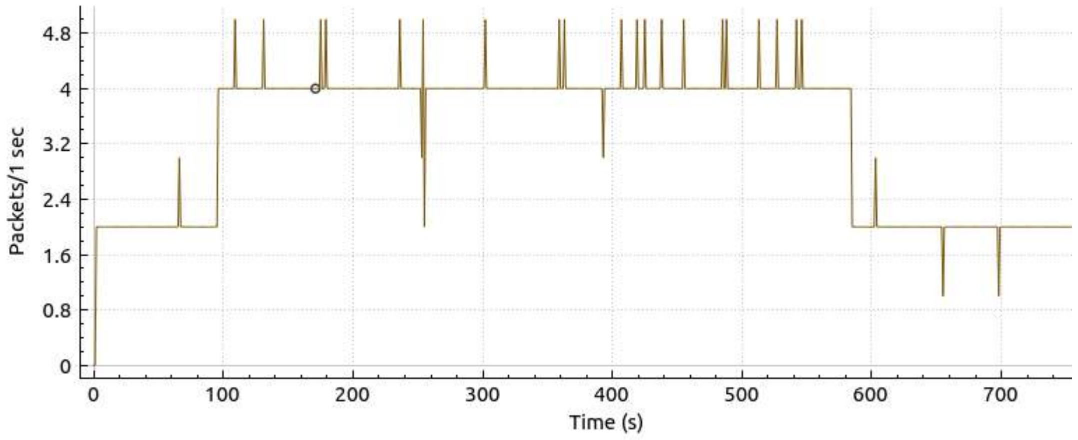


Figure 4.26: Effecting of prevention mechanism after the flooding

After the monitoring process would notify that there is a huge amount of traffics that need to prevent or drop it. Then after the dropping process it the performance of the network was enhance as it show in figure the curve was going to down that mean there is high throughput on the network to provided services.

Verification of iptables Rule sets:

To check your set up iptable rules we must use the v option for a verbose output:

```
# sudo iptables -vL
```

If you want to log the dropped packets on network interface ens32, you can use the following command:

```
# iptables -A INPUT -i ens32 -j LOG --log-prefix " Recycle Bin:"
```

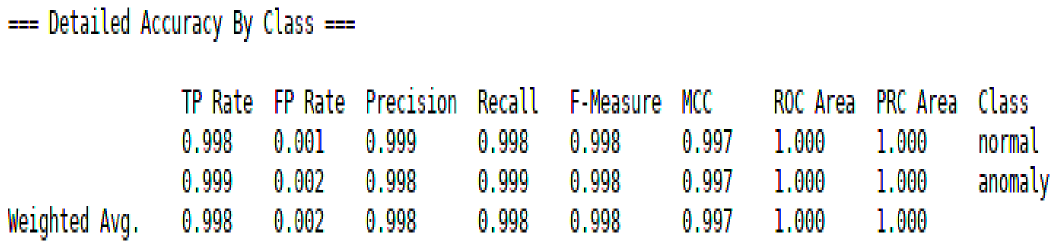
The messages are logged in /var/log/messages and you can search for them with:

```
# grep " Recycle Bin:" /var/log/messages.
```

4.7 Evaluation of proposed model

The proposed model of enhanced hybrid intrusion detection and prevention system that based on signature and anomaly based detection of decision tree algorithm will work more efficient by make deeply packet inspection for detecting and preventing the flooding attacks.

In anomaly based detection technique ,the proposed decision tree algorithm make enhancement of accuracy of detection rate to became 99.8 % and it reduce the values of true and false positive to these values for normal and anomaly traffics as it see in figure 4.27:



```
=== Detailed Accuracy By Class ===

                TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                0.998    0.001    0.999      0.998    0.998      0.997    1.000     1.000     normal
                0.999    0.002    0.998      0.999    0.998      0.997    1.000     1.000     anomaly
Weighted Avg.   0.998    0.002    0.998      0.998    0.998      0.997    1.000     1.000
```

Figure 4.27: Results of classification

Table 4.1 contains simple accuracy comparisons between the proposed algorithms and many other algorithms as in table 4.1:

Table 4.1 : Comparison between the previous results

Algorithm	Accuracy rate
Gentic algorithms+ Neural Network	80%
K-Medoids + SVM	91.5
Decision Stump	92.4%
EMA	92%
BAT Algorithm	94%
The Proposed algorithm	99.8%

Chapter V

Conclusion And Future Work

5.1 Conclusion

In this thesis, a new hybrid intrusion detection method that integrates a signature detection model and an anomaly detection model is introduced .The observed data will pass to detection engine and compare it with predefined rules or signature if it's match mean that there is attacks .Then it use decision tree that is based on c4.5 algorithm to build anomaly detection model to classify a traffics according to the behaviors.

The results show that the proposed hybrid intrusion detection system with hybrid approaches of detection are signature based and anomaly based detection through using decision tree of data mining techniques has a ability of detection attacks (e.g : ICMP flooding , SYS flooding , UDP flooding , HTTP flooding and Port scan attacks) with the faster time and higher detection rate that reach to 99.8% and also it reduces the number of false positives on the detection system to 0.001 when it compares with previous result of IDS as it saw in figure 4.27 and table 4.1

Hence, the security of a network can be enhanced through implement the proposed enhanced hybrid intrusion detection system that provide higher accuracy in terms of anomaly detection and faster execution time in terms of signature based detection. The performance of new hybrid intrusion detection system was tested with NSL-KDD Cup99 Dataset.

In additional to that, the thesis aimed to build protection mechanism through using iptables firewall to define rules that prevent the tested attacks as it saw in figure 4.26.

5.2 Future Work

This project has studies and implementation for enhancing the hybrid intrusion detection system by applying signature and anomaly based detection vi using decision tree based on C4.5 algorithm and reach to acceptable results.

However, there are still several areas where need more work to researches and can be done in future such as try to build intrusion detection systems using different data mining algorithms and to punish mark these systems using various types of flooding attacks or distributed denial of service attacks.

In additional to that , the proposed mode of the enhanced hybrid system can be implemented in various types of the network such as in wireless networks , multi-protocol label switching networks and software defined networks .

Also, there are many protection system (e.g. pefsense) that can be used to protect the resources for unauthorized activities and can be integrated with the proposed enhanced hybrid IDS.

References

[1] William Stallings, “Network security essentials: application and standards”, in Pearson Education, -Inc person highered., ISBN 10: 0-13-610805-9 ISBN 13: 978-0-13-610805-4, 4 edition ,2011.

[2] https://www.snort.org/ , last access : 5 Dec 2017.

[3] From interview of IT employee in the organization, and related papers.

[4] Roberto, luigi,”Intrusion Detection Systems” in Springer, ISBN-13 9780387772 653, 2008th Edition, 2008 .

[5] Stephen Northcutt and Judy Novak, “Network Intrusion Detection”, ISBN-1 : ISBN: 978-0735712652, 3rd Edition, September 2002.

[6] Alan Schwartz, Gene Spafford, Simson Garfinkel ,”Practical UNIX and Internet Security “in O'Reilly Media, Inc , ISBN: 0596003234,3rd Edition ,February 2003 .

[7] Ali A. Ghorbani , Wei Lu , Mahbod Tavallaee , “Network Intrusion Detection and Prevention” by Springer ,ISBN: 978-1461424741, ASIN: 1461424747 ,2010 edition ,February 25, 2012.

[8] Carl Endorf,Gene Schultz,Jim Mellander ,”Intrusion Detection & Preventio**n”** in Tata McGraw-Hill Education Pvt. Ltd. , ISBN 13: 9780070616066 ,first edition , 2006 .

[9] B.Santos Kumar, T.Chandra Sekhara Phani Raju, M.Ratnakar, Sk.Dawood Baba, N.Sudhaka ,“Intrusion Detection System- Types and Prevention” ,in International Journal of Computer Science and Information Technologies . Vol. 4 (1), 77 - 82.2013.

[10] J.David Irwin, Chwan-Hwa Wu,”Introduction to Computer Networks and Cybersecurity” ,in CRC Press , ISBN: 9781466572133 , fourth edition , April 2016.

[11] David J. Marchette,”Computer Intrusion Detection and Network Monitoring: A Statistical Viewpoint” in Springer, ISBN: 9780387952819, 1st Edition, 2001.

[12] Rafeeq Rehman, Refeeq Rehman,”Intrusion Detection with Snort: Advanced Ids Techniques Using Snort, Apache, MySQL, PHP, and Acid” in Prentice Hall , ISBN:9780131407336 , 2nd Edition ,2003.

[13] Northcutt, Stephen, “Network Intrusion Detection”, in Pearson Education, ISBN: 9780735712652, 3rdEdition, 2002.

[14] Christopher Kruegel, Fredrik Valeur, Giovanni Vigna ,”Intrusion Detection and Correlation” , in Springer-Verlag New York Inc ,ISBN**:**9781441936240 , 1st Edition , 2010 .

[15] Jeffrey Pusluns; Jay Beale; Brian Caswell; James C. Foste,"Snort 2.0 - Intrusion Detection", in Elsevier Science & Technology Books, ISBN: 9781931836746, 2ndedition, 2003.

[16] Jiawei Han, Micheline Kamber, Jian Pei,"Data mining: concepts and techniques", Morgan Kaufmann, ISBN: 978-0-12-381479-1 ,3rd Edition, 2011.

[17] Ian Witten Eibe Frank,"Data Mining Practical Machine Learning Tools and Techniques", in Morgan Kaufmann, ISBN: 9780080477022, 2nd Edition, 8th June 2005.

[18] P.Akshaya,"Intrusion detection system using machine learning approach" in International journal of engineering and computer science, ISSN: 2319-7242, Vol. 5, Issue 10, Oct. 2016.

[19] L. Khalvati , M. Keshtgary and N. Rikhtegar , "Intrusion detection based on a novel hybrid learning approach " , in Journal of AI and data mining , Vol. 6, No 1,157-162, 2018.

[20] Pradhnya Kamble, R. C. Roychaudhary , "Discrimination Prevention in Data Mining for Intrusion and Crime Detection", in International Research Journal of Engineering and Technology , Vol. 4 , 2016.

[21] Özge cepheli1, Saliha Büyükçorak, and Güneg Karabulut Kurt," Hybrid intrusion detection system for DDoS Attacks" in Journal of electrical and computer engineering, Vol. 2016, 2016.

[22] Aliya Ahmad , Bhanu Pratap Singh Senga ." Instruction detection system based on support vector machine using BAT algorithm" in international journal of computer applications, Vol. 158 – No 8, January 2017.

[23] Matthew Helmke,"Ubuntu Unleashed", Addison-Wesley Professional, ISBN: 9 78-0134985466, 13th Edition, August 19, 2018.

[24] Caswell Jay Beale Andrew Bake, "Snort Intrusion Detection and Prevention Toolkit", in Syngress , ISBN: 9780080549279 ,1st Edition, 27th March 2007.

[25] http://www.snort.org/docs/snort_manual/ -last access on 22/2/2018.

[26] https://www.cs.waikato.ac.nz/~ml/weka/documentation.html - last access on 5/4/2018.

[27] https://www.wireshark.org/docs/wsug_html_chunked/ChapterIntroduction.html -last access on 6/5/2018.

[28] https://www.unb.ca/cic/datasets/nsl.html -last access on 12/6/2018.

Appendix A

I. Rules of signatures based detection

$Id: local.rules

# LOCAL RULES

# This file intentionally does not come with signatures, I add these signatures manually to work together to enhance the efficiency of the detection

# This rules add more accuracy to the intrusion detection system by make deep packet section by consider the size of packets.
# Nmap port scanning rule:

# These rules will listen to the messages/packets that are based on TCP protocol when it use nmap for scanning process:

alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"SCAN nmap XMAS"; flow:stateless; flags:FPU,12; reference:arachnids,30; classtype:attempted-recon; sid:1228; rev:7;)

# ICMP Flooding rules:

# These rules will listen to the messages that are based on icmp protocols and it display different types of messages:

alert icmp any any -> $HOME_NET any (msg:"ICMP test detected";GID:1; classtype:icmp-event; sid:10000001; rev:001;)
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP Echo Reply"; icode:0; itype:0; classtype:misc-activity; sid:408; rev:5;)
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP Echo Reply undefined code"; icode:>0; itype:0; classtype:misc-activity; sid:409;)
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP Large ICMP Packet"; dsize:>800; reference:arachnids,246; classtype:bad-unknown; sid:499; rev:4;)
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP PING"; icode:0; itype:8; classtype:misc-activity; sid:384; rev:5;)

# TCP Rules:

# These rules will listen to the messages that are based on TCP protocols and it display different types of messages:

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

alert udp $EXTERNAL_NET any <> $HOME_NET 0 (msg:"BAD-TRAFFIC udp port 0 traffic"; reference:bugtraq,576; reference:cve,1999-0675; reference:nessus , 10074 ; classtype:misc-activity; sid:525; rev:9;)
alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"BAD-TRAFFIC data in TCP SYN packet"; flow:stateless; dsize:>6; flags:S,12; reference :url , ww w .cert .org/incident_notes/IN-99-07.html;classtype:misc-activity; sid:526; rev:11;)
alert ip any any <> 127.0.0.0/8 any (msg:"BAD-TRAFFIC loopback traffic"; reference:url,rr.sans.org/firewall/egress.php; classtype:bad-unknown; sid:528; rev:5;)
alert tcp any any -> 172.16.34.0/24 any (flags: SF; msg: "SYNC-FIN packet detected";)

# UDP Rules:
# These rules will listen to the messages that are based on UDP protocols and it display different types of messages when it greater than particular size:
alert udp $EXTERNAL_NET any -> $HOME_NET any (msg:"BAD-TRAFFIC data in UDP packet"; flow:stateless; dsize:>40; reference :url,www.cert.org/ incident _notes /IN-99-07.html; classtype:misc-activity; sid:526; rev:12;)
alert udp $EXTERNAL_NET any -> $HOME_NET any (msg:"MISC Large UDP Packet"; dsize:>4000; reference:arachnids,247; classtype:bad-unknown; sid:521; rev:2;)

# IP Rule:
#This rule will listen to the message that is based on ip protocols and it display Large size IP packet detected messages when the size of message is greater than 6000 :
alert ip any any -> 172.16.34.0/24 any (dsize: > 6000; msg: "Large size IP packet detected";).

# HTTP Flooding Rules :
# This rule will listen to the message that is use http protocols and it display GET matched message when it matched with the get method of the request http message:
alert tcp any any -> 172.16.34.0/24   80(content: "GET"; msg: "GET matched";)
alert tcp $EXTERNAL_NET any -> $HOME_NET 22 (msg:"HTTP Packets Detected"; sid:10000003;)

Appendix B

ii.Anomaly Detection code

```
  /* java code for decision tree based on C4.5 algorithm:
package weka.classifiers.trees;
import java.util.*;
import weka.core.*;
import weka.classifiers.*;
public class J48consolidated extends AbstractClassifier implements OptionHandler,
Drawable, Summarizable ,Matchable, Sourcable, AdditionalMeasureProducer ,
PartitionGenerator WeightedInstancesHandler , TechnicalInformationHandler, {
    /** for serialization */
    static final long serialVersionUID = -2177331683936444555L;
/** The default value set for the percentage of coverage estimated necessary to
adequately cover **/
  * the examples of the original sample with the set of samples to be used in the
consolidation process */
    /** The decision tree    */
    protected ClassifierTree m_root;
/* protected means that the member can be accessed by any class in the same
package or in another package */
/** Use MDL correction? */
    protected boolean m_useMDLcorrection = true;
    /** Unpruned tree? */
    protected boolean m_unpruned = false;
    /** Collapse tree? */
    protected boolean m_collapseTree = true;
    /** Confidence of tree level */
    protected float m_CF = 0.25f;
    /** Minimum number of instances */
    protected int m_minNumObj = 2;
        /**
      * Determines whether probabilities are smoothed using Laplace correction when
      * predictions are generated
      */
    protected boolean m_useLaplace = false;
    /** Use reduced error pruning? */
    protected boolean m_reducedErrorPruning = false;
    /** Number of folds for reduced error pruning. */
    protected int m_numFolds = 3;
    /** Binary splits on nominal attributes? */
    protected boolean m_binarySplits = false;
    /** Subtree raising to be performed? */
    protected boolean m_subtreeRaising = true;
    /** Cleanup after the tree has been built. */
    protected boolean m_noCleanup = false;
    /** Random number seed for reduced-error pruning. */
    protected int m_Seed = 1;
    /** Do not relocate split point to actual data value */
    protected boolean m_doNotMakeSplitPointActualValue;
    /**
    Returns a string describing classifier
      public String globalInfo() {
        return "Class for generating a pruned or unpruned C4.5 decision tree. For more "
          + "information, see\n\n" + getTechnicalInformation().toString(); }
// Returns default capabilities of the classifier.
      @Override
/ *Override is used to define the behaviors that specific to the other class or it mean
the sublass use the methods of superclass */
    public Capabilities getCapabilities() {
        Capabilities result;
// will want to resrve space on memory for the object
        p = new Capabilities(this);
        result.disableAll();
        // attributes
        p.enable(Capability.NOMINAL_ATTRIBUTES);
        p.enable(Capability.DATE_ATTRIBUTES);
        p.enable(Capability.MISSING_VALUES);
        p.enable(Capability.NUMERIC_ATTRIBUTES);
        p.enable(Capability.MISSING_CLASS_VALUES
    // class
        p.enable(Capability.NOMINAL_CLASS););
        // instances
        p.setMinimumNumberInstances(0);
      return result;}
    /**
      * Generates the classifier.
      * @param instances the data to train the classifier with
      */
    @Override
    public void buildClassifier(Instances instances) throws Exception {
        getCapabilities().testWithFail(instances);
        ModelSelection mSelection;
        if (m_binarySplits) {
            modSelection = new BinC45MSelection(m_minNumObj, instances,
                m_useMDLcorrection, m_doNotMakeSplitPointActualValue);
        } else {
            modSelection = new C45MSelection(m_minNumObj, instances,
                m_useMDLcorrection, m_doNotMakeSplitPointActualValue); }
        if (!m_reducedErrorPruning) {
            m_root   =   new   C45PruneableClassifierTree(modSelection,   !m_unpruned,
m_CF,
            m_subtreeRaising, !m_noCleanup, m_collapseTree);
        } else {
            m_root = new PruneableClassifierTree(modSelection, !m_unpruned,
            m_root.buildClassifier(instances
            m_numFolds, !m_noCleanup, m_Seed);        });
        if (m_binarySplits) {
            ((BinC45ModelSelection) modSelection).cleanup();
        } else {
            ((C45ModelSelection) modSelection).cleanup(); } }
```

```
  /**
    * Classifies an instance.
    * @param instance the instance to classify
    */
  @Override
  public double classifyInstance(Instance instance) throws Exception {
    return m_root.classifyInstance(instance); }
  /**
// Returns class probabilities for an instance.
      * @param instance the instance to calculate the class probabilities for
    * @return the class probabilities
    * @throws Exception if distribution can't be computed successfully
    */
  @Override
  public final double[] distributionForInstance(Instance instance)
    throws Exception {
    return m_root.distributionForInstance(instance, m_useLaplace);   }
// Returns the type of graph that is used
    @Override
  public int graphType() {
    return Drawable.TREE;   }
// Returns graph describing the tree.
  * @return the graph describing the tree
    * @throws Exception if graph can't be computed
    */
  @Override
  public String graph() throws Exception {
    return m_root.graph();   }
// Returns tree in prefix order.
   @Override
  public String prefix() throws Exception {
    return m_root.prefix(); }
  /**
```

```
    * Returns tree as an if-then statement.
      * @return the tree as a Java if-then type statement
    * @throws Exception are used if something goes wrong
    */
   @Override
   public String toSource(String className) throws Exception {
      StringBuffer[] source = m_root.toSource(className);
      return "class " + className + " {\n\n"+ "    public static double classify(Object[]
i)\n" + "        throws Exception {\n\n" + "    double p = Double.NaN;\n"              +
source[0] // Assignment code + " return p;\n" + "    }\n" + source[1 + "}\n"; }
// Support code
   /**
    * Returns an enumeration describing the available options.
   * Valid options are:
     * -U: Use unpruned tree
    * -L : * Do not clean up after the tree has been built.
    * -A : If set, Laplace smoothing is used for predicted probabilites.
   * -M number: it's Set minimum number of instances per leaf. (Default: 2)
    * -C confidence <br>
    * Set confidence threshold for pruning. (Default: 0.25)
    * -N number: to Set number of folds for reduced error pruning. One fold is used
    * Use reduced error pruning. No subtree raising is performed. as the
    * pruning set. (Default: 3)
       * -B: to Use binary splits for nominal attributes.
     * -S : for    Don't perform subtree raising.
   return an enumeration of all the available options.
    */
   @Override
   public Enumeration<Option> listOptions() {
      Vector<Option> newVector = new Vector<Option>(13);
      newVector.addElement(new Option("\tUse unpruned tree.", "U", 0, "-U"));
      newVector.addElement(new Option("\tSet confidence threshold for pruning.\n"
         + "\t(default 0.25)", "C", 1, "-C <pruning confidence>"));
```

```
        newVector.addElement(new Option("\tSet number of folds for reduced error\n"
          + "\tpruning. One fold is used as pruning set.\n" + "\t(default 3)", "N",
          1, "-N <number of folds>"));
        newVector.addElement(new Option("\tDo not collapse tree.", "O", 0, "-O"));
        newVector.addElement(new Option("\tUse binary splits only.", "B", 0, "-B"));
        newVector.addElement(new Option("\tDo not perform subtree raising.", "S", 0,
"-S"));
        newVector.addElement(new Option( "\tSet minimum number of instances per
leaf.\n" + "\t(default 2)", "M", 1, "-M <minimum number of instances>"));
        newVector.addElement(new Option("\tUse reduced error pruning.", "R", 0,
"-R"));
        newVector.addElement(new Option("\tDo not use MDL correction for info gain
on numeric attributes.", "J", 0, "-J"));
        newVector.addElement(new Option( "\tSeed for random data shuffling (default
1).", "Q", 1, "-Q <seed>"));
newVector.addElement(new Option( "\tLaplace smoothing for predicted
probabilities.", "A", 0, "-A"));
        newVector.addElement(new Option("\tDo not use MDL correction for info gain
on numeric attributes.", "J", 0, "-J"));
        newVector.addElement(new Option( "\tSeed for random data shuffling (default
1).", "Q", 1, "-Q <seed>"));
        newVector.addElement(new Option( "\tDo not clean up after the tree has been
built.", "L", 0, "-L"));
        newVector.addElement(new Option("\tDo not make split point actual value.",
"-doNotMakeSplitPointActualValue", 0, "-doNotMakeSplitPointActualValue"));
        newVector.addAll(Collections.list(super.listOptions()));
        return newVector.elements(); }
  @Override
    public void setOptions(String[] options) throws Exception {
        // Other options
        String minNumString = Utils.getOption('M', options);
        if (minNumString.length() != 0) {
          m_minNumObj = Integer.parseInt(minNumString);
        } else {
          m_minNumObj = 2;    }
        m_binarySplits = Utils.getFlag('B', options);
        m_useLaplace = Utils.getFlag('A', options);
        m_useMDLcorrection = !Utils.getFlag('J', options);
        // Pruning options
        m_noCleanup = Utils.getFlag('L', options);
        m_unpruned = Utils.getFlag('U', options);
        m_collapseTree = !Utils.getFlag('O', options);
        m_doNotMakeSplitPointActualValue = Utils.getFlag(
        m_subtreeRaising = !Utils.getFlag('S', options);
"doNotMakeSplitPointActualValue", options);
        if ((m_unpruned) && (!m_subtreeRaising)) {
          throw new Exception( "Subtree raising doesn't need to be unset for unpruned
tree!");}
        m_reducedErrorPruning = Utils.getFlag('R', options);
        if ((m_unpruned) && (m_reducedErrorPruning)) {
          throw new Exception( "Unpruned tree and reduced error pruning can't be
selected "    + "simultaneously!");}
        String confidenceString = Utils.getOption('C', options);
        if (confidenceString.length() != 0) {
          if (m_reducedErrorPruning) {
            throw new Exception("Setting the confidence doesn't make sense "+ "for
reduced error pruning.");
          } else if (m_unpruned) {
            throw new Exception(
"Doesn't make sense to change confidence for unpruned " + "tree!");
          } else {
            m_CF = (new Float(confidenceString)).floatValue();
            if ((m_CF <= 0) || (m_CF >= 1)) {
              throw new Exception(
"Confidence has to be greater than zero and smaller " + "than one!"); } }
        } else {
            m_CF = 0.25f; }
        String numFoldsString = Utils.getOption('N', options);
        if (seedString.length() != 0) {
            m_Seed = Integer.parseInt(seedString);
        } else {
            m_Seed    = 1; }
        super.setOptions(options);
/* super is keyword that is used inside a subclass to call a methods from super-class.
        Utils.checkForRemainingOptions(options);    }
        if (numFoldsString.length() != 0) {
            if (!m_reducedErrorPruning) {
                throw new Exception("Setting the number of folds"      + " doesn't make
sense if"    " reduced error pruning is not selected.");    }
  else {
                m_numFolds = Integer.parseInt(numFoldsString);    }
        } else {
            m_numFolds = 3;        }
        String seedString = Utils.getOption('Q', options);
          Gets the current settings of the Classifier.
    @Override
    public String[] getOptions() {
        Vector<String> options = new Vector<String>();
        if (!m_collapseTree) {
                options.add("-O");        }
      if (m_noCleanup) {
                options.add("-L");    }
        if (m_unpruned) {
                options.add("-U");    }
  else {
            if (!m_subtreeRaising) {
                    options.add("-S");      }
            if (m_reducedErrorPruning) {
                    options.add("-R");
                options.add("-N");
                options.add("" + m_numFolds);
                options.add("-Q");
                options.add("" + m_Seed);
            } else {
                options.add("-C");
                options.add("" + m_CF); } }
        if (m_binarySplits) {
              options.add("-B");        }
        options.add("-M");
        options.add("" + m_minNumObj);
        if (m_doNotMakeSplitPointActualValue) {
              options.add("-doNotMakeSplitPointActualValue");        }
        if (m_useLaplace) {
              options.add("-A");        }
        if (!m_useMDLcorrection) {
              options.add("-J");        }
        Collections.addAll(options, super.getOptions());
        return options.toArray(new String[0]); }
    public String seedTipText() {
        return "The seed used for randomizing the data " + "when reduced-error pruning
is used.";
public String useMDLcorrectionTipText() {
        return "Whether MDL correction is used when finding splits on numeric
attributes.";    }
public boolean getUseLaplace() {
    return m_useLaplace;      }
  public void setUseLaplace(boolean newuseLaplace) {
    m_useLaplace = newuseLaplace; }
      public int getSeed() {
        return m_Seed; }
  public void setSeed(int newSeed) {
        m_Seed = newSeed; }
  public String useLaplaceTipText() {
      return "Whether counts at leaves are smoothed based on Laplace."; }
//Get the value of useMDLcorrection.
  public boolean getUseMDLcorrection() {
      return m_useMDLcorrection;}
// Set the value of useMDLcorrection.
        public void setUseMDLcorrection(boolean new useMDLcorrection) {
      m_useMDLcorrection = newuseMDLcorrection; }
// Returns a description of the classifier.
     @Override
   public String toString() {
      if (m_root == null) {
         return "Not able to build classifiert"; }
      if (m_unpruned) {
         return    "J48consolidated         unpruned    tree\n------------------\n"    +
m_root.toString();
      } else {
         return    "J48    consolidated    pruned    tree\n------------------\n"    +
m_root.toString(); } }
// Returns the number of leaves
     public double measureNumLeaves() {
      return m_root.numLeaves(); }
// Returns a superconcise version of the model
public String toSummaryString() {
      return "Number of leaves: " + m_root.numLeaves() + "\n"+ "Size of the tree: " +
m_root.numNodes() + "\n"; }
// Returns the size of the tree
   public double measureTreeSize() {
      return m_root.numNodes();
public double measureNumRules() {
      return m_root.numLeaves(); }
   @Override
   public Enumeration<String> enumerateMeasures() {
        Vector<String> newVector = new Vector<String>(3);
        newVector.addElement("measureTreeSize");
        newVector.addElement("measureNumRules");
        newVector.addElement("measureNumLeaves");
        return newVector.elements();}
    @Override
    public double getMeasure(String additionalMeasureName) {
        if (additionalMeasureName.compareToIgnoreCase("measureNumRules") == 0)
{
            return measureNumRules();
        } else if (additionalMeasureName.compareToIgnoreCase("measureTreeSize")
== 0)
{
          return measureNumLeaves();
        } else {
{
            return measureTreeSize();
        } else if (additionalMeasureName.compareToIgnoreCase("measureNumLeaves")
== 0) else{
            throw new IllegalArgumentException(additionalMeasureName + " not
supported (j48Consolodated)"); }}
  public String unprunedTipText() {
        return "Whether pruning is performed.";    }
  public boolean getUnpruned() {
        return m_unpruned;}
  public void setUnpruned(boolean v) {
        if (v) {
            m_reducedErrorPruning = false; }
        m_unpruned = v;}
  public String collapseTreeTipText() {
        return "Whether parts are removed that do not reduce training error."; }
//* Get the value of collapseTree
    public boolean getCollapseTree() {
        return m_collapseTree;    }
// Set the value of collapseTree.
    public void setCollapseTree(boolean v) {
        m_collapseTree = v; }
      public String confidenceFactorTipText() {
        return "The confidence factor used for pruning (smaller values incur "+ "more
pruning).";}
    public float getConfidenceFactor() {
        return m_CF; }
    public void setConfidenceFactor(float v) {
        m_CF = v;    }
// Returns the tip text for this property
          public String minNumObjTipText() {
        return "The minimum number of instances per leaf.";    }
// need to Get the value of minNumObj.
      public int getMinNumObj() {
        return m_minNumObj;    }
// Set the value of minNumObj.
    public void setMinNumObj(int v) {
        m_minNumObj = v; }
    public String reducedErrorPruningTipText() {
        return "Whether reduced-error pruning is used instead of C.4.5 pruning.";    }
// Get the value of reducedErrorPruning.
          public boolean getReducedErrorPruning() {
        return m_reducedErrorPruning;    }
// Set the value of reducedErrorPruning. Turns unpruned trees off if set.
      public void setReducedErrorPruning(boolean v) {
        if (v) {
            m_unpruned = false;    }
        m_reducedErrorPruning = v; }
// Returns the tip text for this property
    public String numFoldsTipText() {
        return "Determines the amount of data used for reduced-error pruning. "
```

```
          + " One fold is used for pruning, the rest for growing the tree.";]
// Get the value of binarySplits.
    public boolean getBinarySplits() {
        return m_binarySplits; }
// Set the value of binarySplits.
      public void setBinarySplits(boolean v) {
        m_binarySplits = v;    }
// Returns the tip text for this property
public String subtreeRaisingTipText() {
        return "Whether to consider the subtree raising operation when pruning."; }
// Sets the value of doNotMakeSplitPointActualValue.
        public void setDoNotMakeSplitPointActualValue(
        this.m_doNotMakeSplitPointActualValue                                          =
m_doNotMakeSplitPointActualValue;}
/ Get the value of numFolds.
        public int getNumFolds() {
        return m_numFolds;    }
// Set the value of numFolds.
    public void setNumFolds(int v) {
        m_numFolds = v; }
  // Returns the tip text for this property
        public String binarySplitsTipText() {
        return "Whether to use binary splits on nominal attributes when "+ "building the
trees."; }
// Get the value of subtreeRaising.
  public boolean getSubtreeRaising() {
        return m_subtreeRaising; }
// Set the value of subtreeRaising.
    public void setSubtreeRaising(boolean v) {
    m_subtreeRaising = v; }
// Returns the tip text for this property
        public String saveInstanceDataTipText() {
        return "Whether to save the training data for visualization."; }
```

```
// Check whether instance data is to be saved.
    public boolean getSaveInstanceData() {
      return m_noCleanup;    }
// Set whether instance data is to be saved.
      public void setSaveInstanceData(boolean v) {
      m_noCleanup = v;    }
    public String doNotMakeSplitPointActualValueTipText() {
      return "If true, the split point is not relocated to an actual data value."
         + " This can yield substantial speed-ups for large datasets with numeric
attributes.";    }
      Gets the value of doNotMakeSplitPointActualValue.
      public boolean getDoNotMakeSplitPointActualValue() {
      return m_doNotMakeSplitPoint ActualValue; }
    boolean m_doNotMakeSplitPointActualValue) {
// Returns the revision string.
      @Override
   public String getRevision() {
      return RevisionUtils.extract("$Revision: 14508 $"); }
   // Builds the classifier to generate a partition.
   @Override
   public void generatePartition(Instances data) throws Exception {
      buildClassifier(data); }
// Computes an array that indicates node membership.
      @Override
   public double[] getMembershipValues(Instance inst) throws Exception {
      return m_root.getMembershipValues(inst); }
// Returns the number of elements in the partition.
   @Override
   public int numElements() throws Exception {
      return m_root.numNodes();    }
// Main method for testing this class
      public static void main(String[] argv) {
      runClassifier(new J48Conslidates(), argv); }}
```