ALTINBAS UNIVERSITY

GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

IMPROVING IDS ALERTS TO IMPROVE THE QUALITY OF THE NETWORK SECURITY BY USING DATA MINING TECHNIQUES

M. Sc. Thesis

ISAM KAREEM THAJEEL THAJEEL

ISTNBUL, 2017

IMPROVING IDS ALERTS TO IMPROVE THE QUALITY OF THE NETWORK SECURITY BY USING DATA MINING TECHNIQUES

By

ISAM KAREEM THAJEEL THAJEEL

ALTINBAS UNIVERSITY

Submitted to the Graduate School of Science and Engineering

In partial fulfillment of the requirements for the degree of

Master of Electrical and Computer Engineering

November 2017

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Prof. Dr. Osman Nuri Uçan

Supervisor

Examining Committee Members

Assoc. Prof. Dr. Oğuz BAYAT	(jury)	________________________
Asst. Prof. Dr. Çağatay Aydın	(jury)	________________________
Asst. Prof. Dr. Oguz ATA	(jury)	________________________
Asst. Prof. Dr. Adil Deniz DURU	(jury)	________________________

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Asst. Prof. Dr. Çağatay Aydın

Head of Department

Approval of [Institution] ____/____/____

Assoc. Prof. Dr. Oğuz BAYAT

Director

ACKNOWLEDGEMENT

I would first like to thank my thesis advisor Prof. Dr. Osman Nuri Uçan the dean of the faculty Science and Engineering at Altinbas University. In addition, the door to Dr. AlSaedi office was always open whenever I ran into a trouble spot or had a question about my Thesis or writing. They are consistently allowed this thesis to be my own work but steered me in the right direction whenever they thought I needed it, who they participated in the validation of this thesis project. Without their passionate participation and input, the validation survey could not have been successfully conducted.

I would also like to acknowledge Assoc. Prof. Dr. Oğuz BAYAT the director of the Graduate School of Science and Engineering at Altinbas University as the second reader of this thesis, and I am gratefully indebted to him for his very valuable comments on this thesis.

Finally, I must express my very profound gratitude to my parents, to my brothers, to my sisters and my wife for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

ABSTRACT

IMPROVING IDS ALERTS TO IMPROVE THE QUALITY OF THE NETWORK SECURITY BY USING DATA MINING TECHNIQUE

ISAM KAREEM THAJEEL THAJEEL,

M.S., Electrical and Computer Engineering Department, Altinbas University,

Supervisor: Prof. Dr. Osman Nuri Uçan

Date: November-2017

Intrusion-detection systems have become an increasingly important part of network security. Two types of intrusion detection systems are more common used, misuse and anomaly. Anomaly build a model of what is benign traffic, anything deviating from this will be flagged as malicious activity. Misuse search for pattern or known strings (signatures) within network traffic, any matching traffic will be considered suspicious. However, often normal network traffic produce matches against signatures, creating large amounts of false alarms.

Data mining techniques looks for patterns or relations between records in a large data set. Frequent Itemset is a data mining technique to find frequently occurring items, in an alert generated. In this thesis applied Association Rules as data mining technique to find items of frequently occurring alarms. From these Itemsets we create rules, which provide ability to calculate the threat degree for all these items of each attribute and then extracts the threat degree of each alarms. The proposed system have been evaluated and tested by using DARPA '99 datasets.

In this thesis, proposed a new system to eliminate the duplicate and redundant IDSs alert which result in minimizing the false positive rate. The proposed system is based on two major phases which each phase consists of several sub-phases. The first phase removes duplicated alerts by apply new filtering algorithm prepared for this purpose. The second phase is to reduce false alerts by eliminating the redundant alerts by apply association rules mining frequent itemsets algorithms. The proposed system is evaluated and tested by using five weeks of DARPA 1999 dataset. The results show that the proposed system significantly reduces the false positive alerts by 97.98%. These results demonstrate the system's high ability to reduce very large amounts of false alarms of intrusion detection systems.

Keywords: Network Security, Intrusion Detection System, False Positive Alerts, Data Mining, Alert Evaluation, Threat Degree of Alerts.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Introduction

Over the last two decades, there has been a huge increase in the usage of both network computers as well as internet access in all sectors of life; this has led to a subsequent rise in both external and internal threats to computer networks security. Of particular note is the fact that such attacks have increased rapidly over the last few years. For example, 3 years ago, 200 new threats to networks security per minute were identified [1].

Here it is important to refer to a number of defense security devices and network systems, such as Firewalls, Intrusion-prevention systems, Intrusions-Detections systems (IDSs), anti-virus systems, and many more systems and devices. In recent years, these devices, which consist of diverse layers, have been placed into the infrastructures of the computer networks. Some of these devices are designed to prevent influxes of malicious network traffic from inside and outside of the network by detecting intrusion activity and triggering alerts which enable the network security systems to block said malicious traffic from accessing their destinations.

One of these systems is the Intrusion-Detection system (IDS), which has become more popular in the field of networks security over the last few years. This system looks to detect and prevent attacks against a network system by producing an increased number of alerts. However, extensive use of the IDS system presents a significant problem; indeed, this extensive use increases the frailty of the system, thus meaning that many of the alerts generated by the IDS are False Positives [2, 3, 4]. False positive alerts are alarm messages generated by IDSs which relate to non-malicious traffic activity. The number of these alerts has increased due to rises in both malicious activity and complex network structures; for this reason, it is practically impossible and difficult to verify the validity of each alert [4].

Previous studies have employed different techniques and approaches in order to address the problems of IDS log-files alerts; indeed, addressing said issues is considered extremely significant when it comes to facing these threats, which have compromised computer networks security for more than 10 years. In fact, it has been found that the alerts indicate the identification of an attack or intrusion; these threats will be explained in the following chapters of this thesis. In actual fact, such threats are caused by hackers or intruders, who attack networks security by using different types of threats, including denial of services (DOSs) attacks and malware attacks, etc. [5].

1.2 Background

Intrusions can usually be traced to intruders and hackers who are seeking to access the network systems of companies, governmental institutions, and so on. These attacks can be carried out by sending a DDos, virus, bot, worms, and other types of malware [6, 7].

Intrusion-detection systems (IDSs) are classified as alert detection systems that are an integral part of the infrastructure of network security. The work of these systems involves monitoring incoming packets form outside or inside of the network and identifying said packets based on the intrusive behaviors [16]. The alerts are generated when malicious behaviors are detected. These alerts provide the security analyst with the chance to react instantly to the possible malicious activities. Such alerts are generated by the IDSs when traffic is identified as malicious; in actual fact, however, most of these are not malicious alerts, and are instead called false positive alerts.

Intrusion-Detection systems help in understanding the external threats by providing the systems analyst with information related to said threats; this will be discussed in detail in Chapter 2. The information is collected and arranged as an alarm message when threats activities are detected. The alarms contain this information, which the network systems analyst can use to repair the defect in the network security system [8].

An intrusion-detection system (IDS) can take the form of software or a hardware device, both of which are used for the same purpose, namely to filter network traffic. The information in the network traffic is not directly transmitted into the IDS devices; instead, these devices observe the network traffic by using an out-of-band network interface [6].

The three main processes of the IDS will be illustrated in detail in the next chapter. Prior to this, below is a brief introduction to these components:

1. **Sniffing:** this process is responsible for copying and capturing all the packets which enter the network.
2. **Preprocess:** the main purpose of this process is to classify the packets based on their protocols and prepare said packets for the detection engine.
3. **Detection engine:** this component is responsible for detecting and analyzing the preprocessed packets, so that they can be classified based on rules that exist in the IDS.

Alerts refer to any type of user warning message related to an attack event. When the IDS detects an intruder activity, the system uses these warning messages to inform the security administrator about the suspected activities. The alert information contains multiple lines, each of which contains several attributes of the alerts [10]. There are a number of common attributes which will be present in most logs of the intrusion detection system. The following sentences present a detailed illustration of the common attributes:

- [GID:SID:RID]: this represents the ID of the alert, and consists of three numbers; each number value represents specific information about the alert generated, including:

1. GID: this number denotes the Generator ID, and indicates which component of the IDS is responsible for producing the alert.
2. SID: this number denotes the Signature ID, and illustrates which signature rule has been adopted in determining the event.
3. RID: this number denotes the Revision ID; this indicates which revision of the signature has been used in detecting the event.

- Date: date of the event occurrence.
- Time: time of the event occurrence.
- ID: unique number used as identifier for each generated alert.
- Classification: indicates which classification rule the alert belongs to.
- RulesFileComes: this attribute indicates which rule file the alert comes from. Generally speaking, the rules of IDS exist within "misc.rules".
- Priority: indicates the levels of threat for each alert. There are usually three levels of priority, including: low, medium, and high.
- IP Source: the attacker’s IP address.
- Port Source: the attacker’s port source.

- IP destination: the target's IP address.
- Port destination: the target port's destination.
- Protocol: the protocol type which is used.
- TOS: Type of Service.
- DgmLen: the packet size in bytes.
- IpLen: IP header size in bytes.
- TTL: time to live.

1.3 Problem Description

There are several levels of security within computer networks. The performance quality of networks security depends mainly on the proper deployment of the networks security devices. One of these devices is the intrusion-detection system, which is used to observe network traffic and thus detect malicious activity. When this system detects malicious network traffic, it generates more alerts than are necessary, and so it is clear that this system has limitations [5, 3, and 11]. Said limitations affect the security quality of the computer networks which the IDS is designed to protect.

The main problem with the intrusion detection system is that it generates a vast number of false negative and false positive alarms [4]. These can occasionally be very high in number, sometimes constituting up to 99% of the alerts produced [6,12]. The reason for these problems is not a single fault, but instead a set of different factors. Indeed, of the vast number of alerts, most are not real, and very few of them are generated from real attacks. Consequently, the real attack alerts commonly go unnoticed, as the security administrator disregards reported incidents because the number of alerts has become too high. Finally, this harms the overall efficiency of the entire intrusion detection system, and almost completely destroys the quality of the networks security.

1.4 Motivation and Goals

Recent times have seen a rapid growth in the use of computer networks around whole the world, which has subsequently resulted in a rise in the use of the intrusion-detection system as an integrated and main part of networks security systems at different levels of networks. Given this situation, false positive alarms are becoming a progressively larger problem for these systems. Evaluating an intrusion-detection system alert is a task which requires a great deal of effort. Indeed, the vast number of alerts means that substantial human resources are needed to handle all of the alarms. Some systems administrators usually prefer to ignore and avoid deploying the IDS to tackle networks security, due to overly high rate of false alarms; indeed, this problem means that there is no way to reap the main benefits of the IDS, which does have the ability to support networks security.

In the proposed system, we intend to improve and enhance the performance of the intrusion detection system by utilizing the data mining techniques algorithms to put forth an alternative approach which is capable of reducing the number of false intrusion alerts. The following points illustrate the main objectives of this thesis:

1. To propose a new filtering algorithm which removes the duplicate alerts and which depends on specific attributes as well as the time stamp attributes and other attributes.

2. To improve existing algorithms so that they are more suitable when used with multiple items of a single attribute. These improved algorithms will be capable of calculating the threat degree of each item (TDI) and the threat degree of each alert (TDA) so as to evaluate the TDA values of alerts and to cluster them. All of the steps in this phase aim to remove the redundant alerts.

1.5 Thesis Contributions

At present, IDS analysts still deal with the complexity of analyzing alerts – a complexity which results from the vast number of duplicated and redundant alerts. The tasks faced by analysts are made particularly difficult and complex due to the lack of an evaluation system to identify the degree of threat. Indeed, such a system would allow them to determine the primacy of the threat, depending on the level of said threat. This thesis puts forth a new mechanism, the purpose of which is to solve these problems with IDS alerts; this is achieved by proposing the following:

1. **Filtering Duplicate Alerts (FDA) phase:** This phase involves a system which proposes a new algorithm to filter and reduce false alarms by removing the duplicated alerts through two sub-phases. The first sub-phase involves removing the duplicate alerts based on the similarities between the alerts' attributes. The second sub-phase involves removing the duplicated alerts based on the similarities between the alerts' attributes and the time stamp attributes.

2. **Alert Threat Evaluation (ATE) phase:** This phase consists of three sub-phases; these sub-phases aim to eliminate the redundant alerts. The first sub-phase is designed to make the Eclat algorithm more efficient so that it is suitable for work with multiple items of single attribute. The second sub-phase is concerned with generating rules for itemset and alert degree evaluation. This sub-phase is based on improving the Rules Generate algorithm so that, in addition to generating rules, it can also calculate the threat degrees of alerts implicitly. The last sub-phase involves clustering the alert threat degree values into two groups.

1.6 Terms and definitions

Below are the formal introductions to, and definitions of, the terms used in this thesis [13, 14, and 15]:

Data Mining: involves a set of algorithms which extract the knowledge from a large dataset.

Detection Rate (DR): the detection rate involves dividing the number of instances (True Positive) which the IDS systems detect by the total number of threat instances (intrusion instances) which are found in the particular dataset.

Alert or Alarm: is a set of system notifications which indicate malicious activity. These notifications are generated by the IDSs when an intruder is detected. The objective of these alarms is to inform the security administrator or security analyst that immediate action should be taken to prevent these attacks from accessing their destinations.

Attack: is the actual and real implementation of threats.

Threat or Intrusion: refers to a set of events in a communication network or system which are thought to be events that could potentially lead to an assault on the security policy.

True positive: is a right attack which triggers a detection engine (IDS) to produce an alarm.

False positive: represents normal instances which have been incorrectly classified as malicious activity by the intrusion detection system.

1.7 Thesis Structure

This thesis consists of five chapters. The current chapter presents an introduction to, and a brief background of, the proposed system work.

Chapter 2: presents the current and related background research in the area of reducing false positive IDS alerts.

Chapter 3: describes the methodology of the proposed system, presents a detailed illustration of the architectures of the system, and explains the implementation details of the entire proposed system.

Chapter 4: illustrates the experiments which have been executed to evaluate the proposed system. This chapter provides the evaluation results of each phase of the system, while also discussing the obtained results; this is followed by a comparison of the results with those from the other related systems.

Chapter 5: describes the conclusion of the proposed system work within this thesis and suggests directions for future work.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This chapter comprises two major sections: the first section is Background, while the second is Related Work. The first section presents a detailed analysis of the Intrusion-Detection System, Data Mining Techniques, and Dataset Selection.

The second section, Related Work, covers the alert reduction techniques; this involves a description of all techniques and approaches used for alert reduction purposes.

2.2 Background

This section is divided into main sub-sections, which highlight both the intrusion-detection system, data mining techniques, and datasets that are used in the proposed system.

2.2.1 Intrusion Detection

Intrusion detection involves software or devices which are used to detect intrusions into system security policy [16]. Intrusion detection looks for intrusive activity which differs from normal activity. As such, an intrusion detection system (IDS) is used to achieve the main security goals: integrity, confidentiality, and availability of computer networks and system [17]. The purpose of using an IDS is to complement the existing infrastructure networks security measures. To achieve this, the IDS monitors the network traffic to detect malicious traffic activity and generate alarm notification when the potential intrusion activity is detected. It is for this reason that intrusion detection systems (IDSs) are considered as a main component of computer networks security, and the second line of defense for the networks and systems. There are six main types of intrusions [18], all of which must be identified by IDSs.

- **Masquerade attacks:** this type of intrusion is discovered when there are violations of security restrictions or irregular behavior profiles.

- **Attempted break-ins:** this type of intrusion is detected when there are violations of security restrictions or if there exist irregular behavior profiles.

- **Leakage:** this type of intrusion is flagged up if there is irregular use of system resources.

- **Malicious use:** this type of intrusion is detected when there are irregular behavior profiles, use of special privileges of system resources, or violations of security restrictions.

- **Penetration of the security control system:** this type of intrusion is flagged up if specific activity patterns are observed.

- **Denial of service (DOS):** this type of intrusion is detected following the observation of irregular use of system resources.

2.2.1.1 Intrusion-Detection system based techniques

Based on the actions that are taken, certain scope and detection techniques are used. The intrusion detection can be divided into two main categories: the

Anomaly Detection technique and the Misuse-Detection technique. In actual fact, these system types are not mutually exclusive. More specifically, the IDS systems may consist of both detection techniques together in a single system [19].

(A) Anomaly Detection systems

These systems suppose that the benign traffic differs from intrusive traffic, due to their different characteristics. These systems create a profile which contains the normal traffic, and then identify all traffic which differs from this profile as intrusion traffic. Systems of this type are usually computationally expensive, because of the need to keep these profiles updated with normal behaviors. In addition, due to the method that this system employs to identify the attacks, it is not always possible to provide details on all of the attacks [6]. Anomaly detection systems have the ability to detect currently unknown attacks when they deviate from the type of traffic which is in the profile; this point is considered the main advantage of the anomaly detection system. Nevertheless, this technique has low accuracy, which leads to the generation of a large number of false alarms [20].

(B) Misuse Detection systems

Misuse detection systems are designed to detect attacks by looking for common sequences or patterns within network traffic. Systems which use the misuse technique are referred to as signature systems by most researchers. These systems only have the ability to detect attacks whose signatures are stored in the database within said systems [21]. This database contains the patterns of attacks, which the misuse detection systems depend on to detect the abnormal patterns. The advantage of the misuse-detection systems compared with the anomaly-detection systems is that the misuse-detection systems have the ability to provide the security administrator with reports that contain more details of attacks which have been detected. Despite this, the misuse-detection systems have a fundamental problem, specifically that the signatures must be stored correctly and should contain all possible types of attacks. At the same time, these signatures must not include the normal activity [19, 20].

2.2.1.2 Intrusion-Detection System Based Locations

To identify the categories of intrusion detection systems, it is important to look at their location in the environment and infrastructure of networks security. Besides this, it is important to identify what these systems are trying to protect depending on their location [22]:

- **Host-based IDS (HIDS):** HIDSs usually reside in network hosts or servers as software applications; they are considered the first line of defense when it comes to detecting intruders in single hosts or single servers. HIDSs are more suitable for preventing attacks which pinpoint the vulnerabilities of the servers and operating systems. The data of such systems is collected by inspecting the audit records of application programs and system logs from the operating system. HIDSs analyze this data before it is encrypted, while such a feature is not available with the NIDSs. It has emerged that the HIDSs inspect only the intrusions that target hosts. However, there is a problem with HIDSs, namely that they generate a large number of false alarms. In addition, HIDSs are more time consuming and costly because of the need to provide the service for each host. The main disadvantage of HIDS is that they can be easily overcome by DDoS attacks [23].

- **Network-based IDS (NIDS):** NIDSs reside at the perimeter of the network, usually behind a firewall. NIDSs inspect the packet captures at the transport layer and network layer of the OSI model. NIDSs use either the anomaly detection or signature based detection system techniques**.** NIDSs are very important to the network computer security. As such, most computer networks have at least one IDS which is deployed at the network perimeter [24]. Indeed, the NIDSs contain several sensors that are distributed at different locations across the network perimeter. Because of these features, NIDSs have an increased ability to observe a vast amount of network traffic due to their flexibility and the fact that they can be located at any point in the network. The biggest disadvantage of NIDSs is the fact that they are unable to identify whether or not these attacks have been successful. This disadvantage results in performance failure, which pushes the rate of false positives above 99% [3].

- **Protocol-based IDS (PIDS):** PIDSs are usually installed on network hosts or web servers, and are used to analyze and monitor the encrypted traffic activity if the protocol (e.g. HTTPs) provides it; as such, the PIDS systems have access to extensive knowledge. PIDSs inspect each single set of protocols in order to identify whether one of these is an intrusion activity [73].

- **Stack-based IDS (SIDS):** SIDSs are intrusion detection systems which inspect the network packets that are traveling over the TCP/IP stack [73].

- **Graph-based IDS (GIDS):** these are intrusion-detection systems which are used to identify intrusions that make a connection between many network hosts. GIDSs construct an activity graph which represents the activity and hosts in networks. The graph includes the domain, which is represented as nodes, and the network traffic between them, which is represented as edges. The graph is then compared with the known signature patterns of intrusion or malicious activity; if the patterns match up, then alarms are generated [73].

2.2.1.3 Intrusion-Detection Systems Based Response Methods

The way in which intrusion-detection systems interact with intrusive activity can be divided into two kinds: Reactive and Passive. In particular, the main difference between these two kinds of response from the IDSs is that the reactive response IDSs react automatically to attacks when they detect said attacks; in contrast, the passive response IDSs generate alerts as a report, and this report is sent to the systems analyst, who will take appropriate action to block attacks [25].

a) Reactive Response IDS

Generally speaking, these types of intrusion-prevention systems (IPSs) have the ability to automatically configure security resources in order to prevent intrusive activity; indeed, such activity is blocked automatically if intrusive activity is detected. This leads to the prevention of attacks before they are completed. However, the main problem with these response methods is that produce a high rate of false positives (RFP) by blocking the normal traffic, which subsequently reduces the efficiency of the network systems.

b) Passive Response IDS

This reaction is considered the most concerning when conducting a survey of intrusion-detection systems. These systems are only responsible for generating reports for each threat which they detect. Systems analysts or security administrators can examine these reports later and establish what has actually happened. The drawback of this system is the fact that the analyst responds to the alerts after the attacks have finished, and the attack's destination has already been accessed; if this were not the case, the analyst would have to monitor the alerts log of IDSs 24 hours a day.

2.2.1.4 Intrusion-Detection alerts

Previous sections of this thesis have discussed intrusion-detection systems and highlighted the main problem with these systems, namely the fact that they generate hundreds of alerts each day, depending on various factors. This huge number of alerts generated by IDSs is viewed as a complex and very serious issue. The IDS systems create a vast number of alerts when said systems detect internal or external malicious traffic on the networks which the IDS systems protect. This problem can have adverse effects. Indeed, the excess number of false positive alarms means that handling the real attacks is very difficult for analysts, even verging on impossible. Furthermore, this may cause the system administrator to become confused and begin ignoring or skipping alerts because of the vast number of generated alerts; such a situation also decreases the quality of IDS security. There are certain factors which lead to alert flooding, as seen below:

- Many of the signatures trigger on abnormal or suspect network traffic. These signatures are not actually categorized as intrusive traffic, but are classed as rare events, e.g. overlapping IP fragments, failed logins or the utilization of URGENT bit in the TCP packet header. Such incidents have revealed that alerts which are generated by these signatures do not represent real intrusion activity [26].
- There are many signatures which provide an inadequate illustration of the intrusion, and this leads IDS systems to trigger false alerts indicating that normal traffic is an intrusive attack. IDS systems use simple string matching, instead of employing regular terms, to detect the attacks. Using this approach allows the IDS systems to follow traffic data [28]. Storing true and valid signatures is a difficult job, and the process mostly results in an insufficient number of signatures [27].
- With regard to alerts which flag up low-level incidents, the systems tasked with finding malicious activity should inspect all packets and report on each single event. For the most part, attacks include several levels, which must be satisfied before the attack is complete. Consequently, one attack can trigger several alerts, although it may be that all of these alerts relate to the same attack. These problems usually result from a lack of sufficient descriptions and general overviews of all types of attacks in the IDS systems.

- The network architecture and its design can also result in the generation of alerts. Fragment packets mostly occur with a network that has low MTU[1]. Many IDS systems will trigger alerts for each fragmented packet, thus causing a flood of alerts [16, 29].

2.2.1.5 SNORT as an example of IDSs

The Snort is a prime example of an Intrusion-Detection system; it is an open source signature-based and anomaly-based detection system which is used as a method of analysis to efficiently detect different types of attack intrusions and generate alarms to warn the systems analyst. This system has been created by Roech (Rafeeq, 2003), and is concerned with flexibility, performance and simplicity. Snort is considered a highly feasible and flexible system which works with many types of databases, such as: Oracle, MySql and so on. In addition, it has the ability to work very well with different types of Operating Systems, including: Linux, Windows, and Ms Dos [30].

This system provides the systems analyst with the ability to configure Snort for execution in four modes, namely: packet sniffer mode, inline mode, packet logger mode and NIDS mode [31]. For these reasons, researchers have coined the term snort multi-mode analyzer. The NIDS mode is considered the most important mode, although it is also the most complicated. In actual fact, this mode gives Snort the ability to inspect the network traffic based on sets of user-defined rules; the captured traffic is matched with these sets of rules in order to identify both the benign and malicious traffic. Finally, based on these results, several actions are performed.

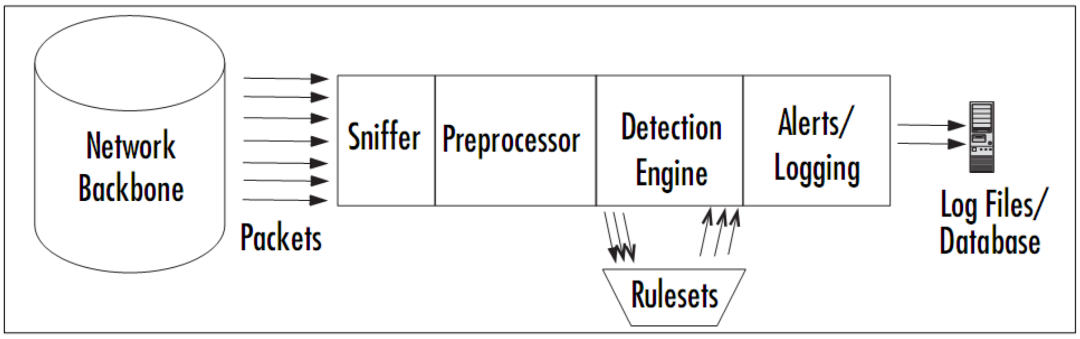
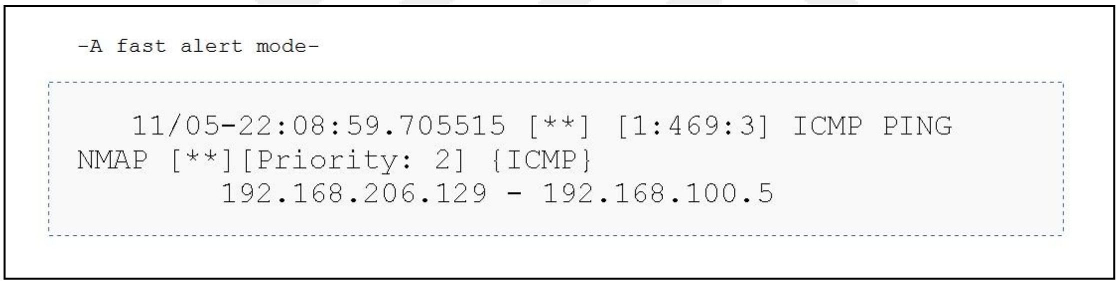


Figure 2.1 Architecture of Snort

As shown in Figure 2.1, the Snort architecture contains four major components, including: the sniffer component, preprocessor component, the detection engine component, and the alerts logging results component [32]. On the other hand, there exist other types of IDSs, such as Bro etc. The Bro is also an open source IDS, although this does not have the ability to work with different types of operating systems; however, the Bro IDS has only been used with Unix-based systems, and thus its use has been limited [33]. In contrast, the Snort is more widely used by researchers around the world.

Snort provides two options which allow users to select the appropriate mode of alert. Indeed, Snort generates two types of alerts: fast alert mode and full alert mode based on the choice made by the user. The significant difference between these two alert modes is that the fast mode contains only the alert message information, such as: time stamp, IP address of source and IP address of destination, as shown in Figure 2.2, which illustrates the components of the fast alert mode.

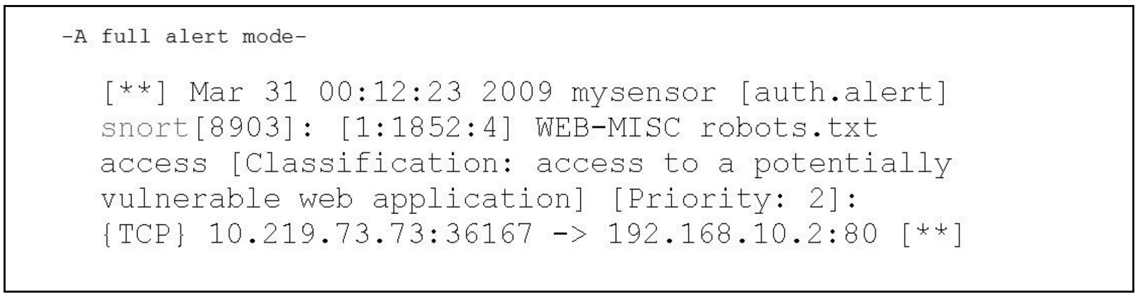


```
-A fast alert mode-

   11/05-22:08:59.705515 [**] [1:469:3] ICMP PING
NMAP [**][Priority: 2] {ICMP}
        192.168.206.129 - 192.168.100.5
```

Figure 2.2 Fast alert mode components

On the other hand, the full mode contains the alert message information, as well as the packet header information; this information generally includes the fast mode components, in addition to extra alert features, including: length of IP packet, length of IP header. These features can be seen in Figure 2.3, which illustrates all of the alert mode components [30].



```
-A full alert mode-

  [**] Mar 31 00:12:23 2009 mysensor [auth.alert]
  snort[8903]: [1:1852:4] WEB-MISC robots.txt
  access [Classification: access to a potentially
  vulnerable web application] [Priority: 2]:
  {TCP} 10.219.73.73:36167 -> 192.168.10.2:80 [**]
```

Figure 2.3 Full alert mode components

The proposed system will be more comfortable and flexible, meaning it can work with alert logs files from different types of IDS. As such, in this thesis uses only the full alert mode, as it mostly contains attributes which are similar to those of other logs files from IDSs.

2.2.2 Data Mining

Generally speaking, data mining is the process of extracting or mining and analyzing knowledge from enormous databases. This process is run by automatic or semi-automatic tools in order to extract new data which is more meaningful [34, 35, and 36]. The main functionalities of data mining are used to uncover hidden information by identifying the types of patterns and rules through establishing the unsuspected relationship between customer behavior and variables that are unobserved in the database; the purpose of this is to introduce data which is more understandable and useful for the user [37].

Typically, the data mining techniques have been categorized into two main classes: predictive and descriptive [12, 35 and 12]. Figure 2.3 depicts the data mining techniques within these two categories.

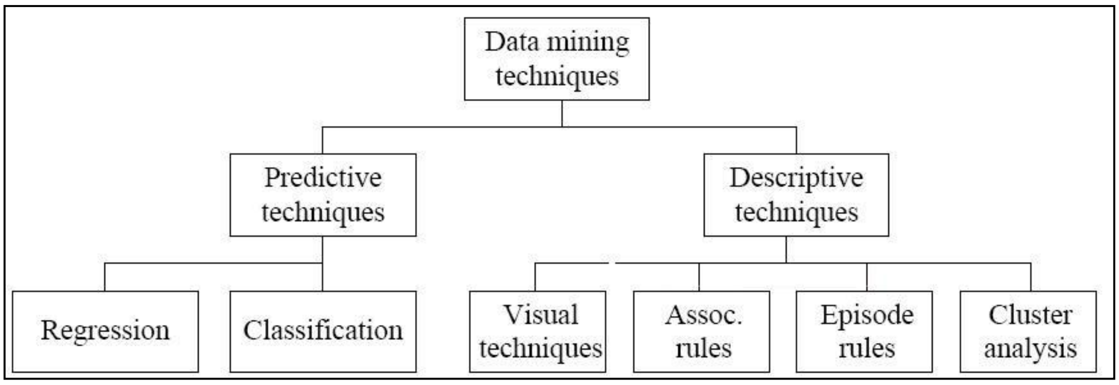


Figure 2.4 The data mining techniques classification

Descriptive techniques are used in the process of mining tasks, the aim of which is to describe the common properties of the data by identifying the relationships and patterns in the database, e.g. sets of rules depending on statistical evidence. These sets of rules are not based on training data. In contrast with the predictive techniques, the descriptive techniques include visual techniques, association rules, episode rules, cluster analysis, and so on. Visual techniques are those techniques which use shapes and colors to represent data; this typically involves employing maps or graphs to visually represent the results. In contrast, the goal of clustering is to classify the data into groups based on calculations of the distance between the items in a cluster; indeed, these distances should be as small as possible. Conversely, the inter-distance among the number of clusters must be as large as possible.

Predictive techniques are used to predict the future value of elements based on training data by using the values of other elements. Generally speaking, predictive techniques are classified into classification and regression categories. Classification uses the categories' data, such as color or name, whereas the order between these types of data is not important. On the other hand, the regression technique uses features, including numeric value, to build a prediction by identifying the relation between this variable value and the value of at least one other variable.

This thesis focuses on **association rules** and **cluster analysis techniques**. Association rules have many advantages, and thus it is very important to use this technique to identify the relationship between the correlation confidence items by mining frequent itemsets to generate sets of itemsets; these sets are the basis for extracting the rules. Indeed, by improving this technique, it is possible to calculate the degree of correlation confidence between the items. The clustering technique also has a significant advantage, and so it is used in the proposed system to cluster the values that are extracted by the association rules techniques.

2.2.2.1 Association Rules

Association Rules represent a central technique in mining knowledge discovery, and involves identifying the correlation relationships or interesting associations among items from large datasets. These relationships, or association rules, are very interesting for decision-makers. The relationship between two items from different datasets is represented as a rule of association; indeed, this is accomplished by utilizing an approach which was proposed in 1993 [35, 40].

An association rule is based on measures of interestingness, which make it possible to extract strong rules which are discovered in databases. These rules are expressed as $A \rightarrow B$, where A and B are a separate set of items and contain a set of items; indeed, this rule means that if an itemset satisfies A, then it must satisfy B. There are two measures of interestingness which are used to define the rules. The first of these measures is the Degree of Support (Supp), which is the ratio of the number of transactions containing both A and B in the entire dataset to the total number of transactions. The second of these measures is the Degree of Confidence (conf), which is the ratio of the number of transactions where A and B appear together to the number of transaction that contain A.

In order for these rules to be called confident rules or strong rules, said rules should satisfy the minimum support threshold (min_sup), as well as the minimum confidence threshold (min_conf) [40]. Support and confidence are different values. Support is used to measure the frequency degree of items in the entire dataset; in other words, this means that an item is more frequent in the dataset if its support value is more than or equal to the min_supp threshold. If this is not the case, then the item is considered less frequent. Conversely, confidence is used to measure the strength of the rules [41].

There are many algorithms which have been proposed to generate association rules. In fact, an association rule is extracted in two steps; the first step involves applying the algorithms of mining frequent itemsets to find the items that are most frequent in a dataset, as well as these items' degree of support. There are three very well-known algorithms, including Apriori, Eclat, and FP-Growth; these algorithms are used to complete the aforementioned actions. The second step involves generating rules from frequent itemsets, which are extracted using the algorithms of the first step by applying the generated rules algorithm.

2.2.2.2 Eclat Algorithm

Association rules are very interesting, and so many researchers have proposed similar algorithms. The Fp-Growth [35] and DICT algorithms [42] are examples of algorithms which have been proposed by such researchers. Indeed, those algorithms which have been proposed after Apriori are thought to differ from it. All of these algorithms need to scan the database several times in order to mine frequent itemsets; this leads to the generation of a huge number of candidate itemsets.

In essence, the Apriori and Eclat algorithms are well-known as the largest families of mining frequent itemset algorithms [43], while the FP-growth is thought to fall within the Eclat family [44]. The Apriori algorithm is based on the breadth-first-search strategy, which generates the K-itemset after scanning the database; this process continues until no frequent itemsets are found. As Apriori rescans the database many times, its efficiency has been affected because of the large number of candidate itemsets which are generated. The Apriori algorithm is based on the horizontal data format.

On the other hand, the Eclat algorithm, proposed by Zaki in 2000, is considered an influential algorithm when it comes to mining frequent itemsets to generate association rules. In actual fact, the Eclat algorithm is based on the depth-first-search strategy, which depends on several factors, including: vertical data format, equivalence classes, lattice theory, intersection and union operation, and so on [43]. The vertical data format can enhance the algorithm's efficiency.

The vertical and horizontal data formats are both representations of data in memory. The vertical data format contains two columns, namely Items and TID_Set. The Items column refers to items in the database, while the TID_Set is the number of transactions which include the items. TID is a unique number identifier of a transaction in a database. These two types of data formats have different levels of efficiency.

The major steps of the Eclat algorithm process are listed below:

1. To generate frequent 1-itemsets, the algorithm scans the entire database.
2. Generate candidate C_1 itemsets form frequent 1-itemsets.
3. Clip non-frequent candidate itemsets of frequent 1-itemsets to obtain frequent 2-itemsets.
4. Generate candidate C_2 itemsets from frequent 2-itemsets.
5. To obtain all frequent 3-itemsets by clipping all non-frequent candidates.

As explained in the above points, firstly the set of frequent 1-itemsets is found. The 1-itsemsets are denoted as C_1, and are used to find C_2 and then generate the set of frequent 2-itemsets; this set is used to find C_3 and then to generate the set of frequent 3-itemsets, and so on. This process carries on until no candidate sets are found.

The Eclat algorithm is identical to the Apriori algorithm, as they both depend on the union operation in order to generate k+1-itemsets by joining two k-itemsets. The only condition is that these two k-itemsets must be the same to be joined. For instance, there are two itemsets for the 2-itemset:

I_{21}= {I1, I4} and I_{22}= {I1, I5}

The first item of the I_{21} itemsets is the same as the first item of the I_{22} itemsets. As such, the I_{21} is joined with I_{22} to generate I_{31} 3-itemset: $I_{21} \wedge I_{22} = I_{31} = \{I1, I4, I5\}$. The support of these C_k candidate itemsets is automatically calculated at the end of each level, by employing two pointers in order to count the matched TIDs in each intersection of the two itemsets.

The Eclat algorithm adopts the equivalence classes, and thus it partitions the search area into multiple various non-overlapping sub-areas whereby the itemsets can be classified into the same category if those itemsets have the same prefix; this means that the process of generating new candidate itemsets can be take place only in the same class. It is clear that the efficiency of generating candidate itemsets is improved by using technology of equivalence classes and reducing the occupation of memory.

2.2.2.3 Association Rules Generation

Once the frequent itemsets from a database (D)'s transactions have been generated, the process moves directly on to generate the intense association rules from these itemsets. The strong rules of association should satisfy both the threshold min_sup and min_conf threshold. The confidence value of each rule is calculated by using the following equation:

$$\text{Confidence } (\mathbf{X} \Rightarrow \mathbf{Y}) = \frac{\boldsymbol{Support}\ (\boldsymbol{X} \cup \boldsymbol{Y})}{\boldsymbol{support}(\boldsymbol{X})} \tag{2.1}$$

where the support (X∪Y) is calculated by counting the number of transactions that contain both items X and Y, and support (X) is calculated by counting the number of transactions that contain item X. Depending on the above equation, the rules generated can resemble the following:

- For each frequent itemset X, generate all non-empty subsets of X.
- For all non-empty subsets a of X, output a rule "x ⇒(X-x)", if

$$[\mathrm{x} \Rightarrow (\mathrm{X} - \mathrm{x})] \geq \min{}_{\_conf}$$

where min_conf: denotes the threshold of minimum confidence vlaue.

The most efficient algorithm for generating association rules from the large frequent set of itemsets is the algorithm put forth by [45, 46, and 47].

2.2.2.4 Clustering

Clustering and classification are both considered essential aspects of data mining. Clustering is used for unsupervised learning, while classification is generally employed as a supervised learning method (some clustering models are used for both the unsupervised and supervised learning methods). The objective of clustering is to describe, while the objective of classification is to predict. Since the objective of clustering is to find a new set of groups, these new groups are more interesting, and their worth is intrinsic (Veyssieres and Plant, 1998) [69]. In classification processes, however, an important part of the evaluation is extrinsic, because the groups must mirror some reference set of the classes.

Clustering involves sorting data instances into groups, and these groups are actually subsets; such groups are clustered using a method whereby similar instances are clustered together into a similar group, while the other instances are sorted into different groups. In this way, the instances are grouped into an effective and functional representation that characterizes the data instances being sampled. In essence, the clustering framework is represented as a set of subsets G= G_1, $G_{2,}$…..,, G_k of S such that: $S = \cup_{x=1}^{K}$, where $G_x \cap G_y = \emptyset$ if $x \neq y$. Thus, any instance in *S* belongs to exactly one group and only one subset.

Most clustering methods define the dissimilarity or similarity between any pair of data instances by using distance measures. It is helpful to refer to the distance between two objects A_i and A_j as: $d(A_{i,} A_j)$. A correct distance measure between two instances should be symmetric and obtain its least value (generally zero) in case of symmetric vectors. The distance measure should satisfy the following attributes in order to be called a metric distance measure:

1. Triangle inequality $d(A_i, A_k) \leq d(A_i, A_j) + d(A_j, A_k)$ where $\forall A_i, A_j, A_k \in S$.
2. $d(A_i, A_j) = 0 \rightarrow A_i = A_j$ where $\forall A_i, A_j \in S$.

There are several kinds of clustering methods, although the main methods are as follows: Hierarchical algorithms (comprising bottom-up-agglomerative and top-down-divisive algorithms) and Partition algorithms (Flat), which contain K-Means and a Mixture of the Gaussian and Spectral Clustering algorithms). The following statements will explain the K-Means algorithm in detail, as the latter is directly related to this thesis.

2.2.2.5 K-Means Algorithm

Generally speaking, the K-Means is an algorithm used to group or to classify data objects, depending on their features/attributes, into the K number of the group. This grouping is carried out by decreasing the distances between the objects and the identical cluster centroid. As such, the objective of K-Means clustering is simply to classify the data.

The more popular distance measures include the Euclidean squared distance, the Euclidean distance, and the City or Manhattan distance, all of which are used to measure the distances between the points. (1) The Euclidean measure is identical to the shortest geometric distance between the data objects, and can be calculated by using the following equation:

$$d = \sqrt{\sum_{i=1}^{N} (A_i - B_i)^2} \tag{2.2}$$

With this said, the faster method for measuring the distance between objects involves squaring the Euclidean distance of the above distance, as follows:

$$d_{sq} = \sum_{i=1}^{N} (A_i - B_i)^2 \tag{2.3}$$

On the other hand, the Manhattan measure (city measure) computes a distance between objects depending on a grid, and is explained in Figure 1.1 below.

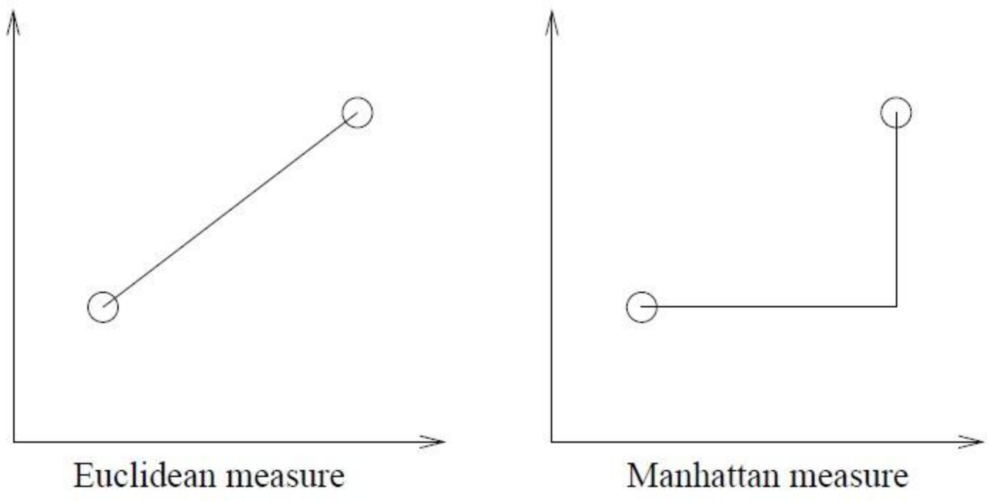
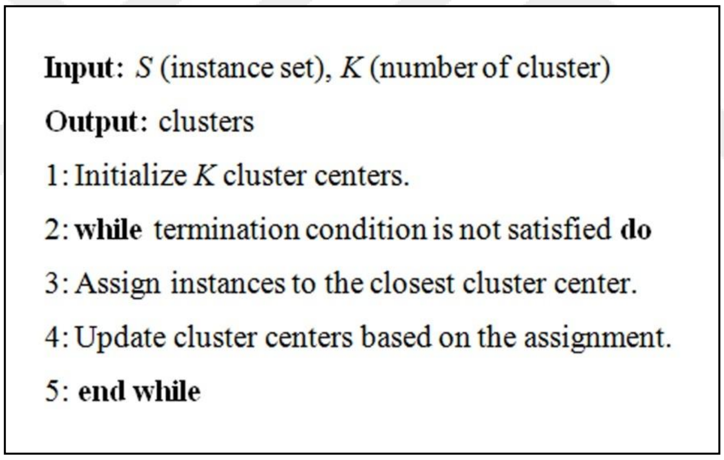


Figure 2.5 Comparisons between Manhattan and Euclidean measures



```
Input: $S$ (instance set), $K$ (number of cluster)
Output: clusters
1: Initialize $K$ cluster centers.
2: while termination condition is not satisfied do
3: Assign instances to the closest cluster center.
4: Update cluster centers based on the assignment.
5: end while
```

Figure 2.6 Pesudocode of K-Means algorithm

As obvious in the above Figure the K-Means algorithm will carry out the three steps below until the convergence Iterate is *stable* (= no object move group) [70, 67]:

Step 1. This step begins with any initial partition of data that groups the objects into k clusters. This can be achieved by following the method set out below:

a. Pick the first k random training examples to be single-element clusters.

b. Assign each element of the remaining examples (N − k) to the nearest centroids of a cluster. After each assignment, recalculate the centroid of the cluster generated

Step 2. Pick each object Cj sequentially and calculate its distance from the centroid of each cluster of the k. Move Cj to another cluster and upgrade the centroid of the cluster generated from Cj and the previous cluster Cj, if Cj does not currently exist in the cluster with the nearest centroid.

Step 3. Repeat step 2 until convergence is achieved, that is until no new assignments.

With a view to understanding the algorithm work, the following example will present a detailed illustration of the steps which make up said algorithm.

Suppose we have several objects (four types of items) and each object has two attributes or features, as shown in Table 2.1 below. The main goal in this example is to group these items into the K=2 group of objects based on the two features (X, Y).

Table 2.1 K-Means algorithm example

Items	(X)	(Y)
A	1	1
B	2	1
C	4	3
D	5	4

Each object item represents one point with two attributes (X, Y) that we can represent as a coordinate in an attribute space.

1. The initial value of centroids: these centroids can be selected randomly from the dataset, or they can be introduced as a threshold by the users. In this example, we will suppose that items A and B are the first and second centroids, with the coordinates C_1 and C_2 in the following form: $C_1(1,1)$, $C_2(2,1)$.

2. Calculate the items-centroids distance: this involves calculating the distance between each item to cluster the centroids. If the Euclidean distance rule is used, then the distance matrix will be at iteration 0, as shown in Table 2.2:

Table 2.2 0-Iteration, calculating items-centroids distance

Items	Objects dis. To C1	Objects dis. To C1
A	$\sqrt{(1-1)^2+(1-1)^2}$=0	$\sqrt{(1-2)^2+(1-1)^2}$ =1
B	$\sqrt{(2-1)^2+(1-1)^2}$ =1	$\sqrt{(2-2)^2+(1-1)^2}$ =0
C	$\sqrt{(4-1)^2+(3-1)^2}$ =3.61	$\sqrt{(4-2)^2+(3-1)^2}$ =2.83
D	$\sqrt{(5-1)^2+(4-1)^2}$ =5	$\sqrt{(5-2)^2+(4-1)^2}$ =4.24

3. Items clustering: each item will be assigned based on the least distance. As shown in the above table, item A is assigned to G_1, and the other items are assigned to G_2.
4. 0-Iteration, determine centroids: in this step, new centroids should be computed. Based on the result in the above tables, the centroid of G_1 will remain the same, since C_1 =$(\frac{1+1}{2},\frac{1+1}{2})$ =(1,1). The centroid of the G_2 will be extracted by calculating the average coordinates among these items' values: C_2=$(\frac{2+4+5}{3},\frac{1+3+4}{3})$ = $(\frac{11}{3},\frac{8}{3})$.
5. 1-Iteration, Items-Centroids distance: similar to the process in step 2, the distance of the new centroids is calculated for all items. As such, the distance will be explained in Table 2.3 below:

Table 2.3 1-Iteration, calculating items-centroids distance

Items	Objects dis. To C1(1,1)	Objects dis. To C1$(\frac{11}{3},\frac{8}{3})$.
A	$\sqrt{(1-1)^2+(1-1)^2}$ =0	$\sqrt{(1-\frac{11}{3})^2+(1-\frac{8}{3})^2}$ =3.14
B	$\sqrt{(2-1)^2+(1-1)^2}$ =1	$\sqrt{(2-\frac{11}{3})^2+(1-\frac{8}{3})^2}$ =2.36
C	$\sqrt{(4-1)^2+(3-1)^2}$ =3.61	$\sqrt{(4-\frac{11}{3})^2+(3-\frac{8}{3})^2}$ =0.47
D	$\sqrt{(5-1)^2+(4-1)^2}$ =5	$\sqrt{(5-\frac{11}{3})^2+(4-\frac{8}{3})^2}$ =1.89

6. 1-Iteration, Items clustering: similar to the process in step 3, each item is assigned to the group based on the least distance between objects and centroids, as shown in Table 2.3. A and B are assigned to G_1, and the other items are assigned to G_2.
7. 2-Iteration, determine centroids: similar to the process in step 4, new centroids are calculated, based on the new group. Where C_1=$(\frac{1+2}{2},\frac{1+1}{2})$=$(\frac{3}{2},1)$, and C_2=$(\frac{4+5}{2},\frac{3+4}{2})$ = $(\frac{9}{2},\frac{7}{2})$.
8. 2-Iteration, Items-Centroids distance: the process in step 2 and step 5 will be repeated again to find the items-centroids distance based on the new centroids, as shown below in Table 2.4:

Table 2.4 2-Iteration, Calculating items-centroids distance

Items	Objects dis. To C1($\frac{3}{2}$,1)	Objects dis. To C1($\frac{9}{2}$,$\frac{7}{2}$).
A	$\sqrt{(1-\frac{3}{2})^2+(1-1)^2}$ =0.5	$\sqrt{(1-\frac{9}{2})^2+(1-\frac{7}{2})^2}$ =4.30
B	$\sqrt{(2-\frac{3}{2})^2+(1-1)^2}$ =0.5	$\sqrt{(2-\frac{9}{2})^2+(1-\frac{7}{2})^2}$ =3.54
C	$\sqrt{(4-\frac{3}{2})^2+(3-1)^2}$ =3.20	$\sqrt{(4-\frac{9}{2})^2+(3-\frac{7}{2})^2}$ =0.71
D	$\sqrt{(5-\frac{3}{2})^2+(4-1)^2}$ =4.61	$\sqrt{(5-\frac{9}{2})^2+(4-\frac{7}{2})^2}$ =0.71

9. 2-Iteration, Items-clustering: as with the processes in step 3 and step 6, again each item is assigned to a new group based on least distance, as shown in the above table. The obtained result shows that the result of clustering in this step is the same as the clustering finding in step 6; indeed, this reveals that no items are moved to either of these two group. As such, no more iteration is needed.

Table 2.5 shows the final grouping result, while Figure 2.6 explains the items' clustering in step 3 and step 6.

Table 2.5 Final grouping result

Items	(X)	(Y)	Group (result)
A	1	1	G1
B	2	1	G1
C	4	3	G2
D	5	4	G2

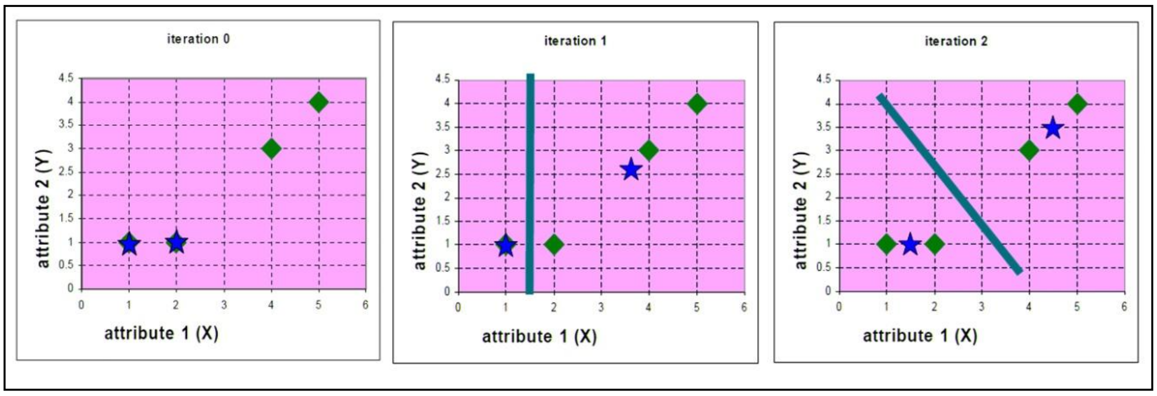


Figure 2.7 K-Means clustering iteration

2.2.3 Dataset Selection

There are several datasets which have been generated to support the evaluation and improvement of the intrusion-detection system performance. These datasets are very important, as they allow the researchers to compare the results of the approaches which have been used by past researchers. There are a number of datasets which are considered most important and are widely used by researchers and those interested in this area, such as: the DARPA 1998-2000 series datasets, KDD 1999 cup datasets, etc. Moreover, there exists another set of databases which are used in intrusion studies, but are less widely employed by researchers, such as: Greenberg (1988), Schonlau and DuMouchel et al. (2001), Moore and Zuev (2005), and many other datasets.

In fact, the Knowledge Discovery and Data Mining KDD 1999 dataset was originally created from the DARPA 1998 dataset, the latter of which was cleaned to prepare the KDD'99 dataset [50]. The DARPA dataset connections were used to deduce 41 potential intrusion attributes. Each one of these attributes was labeled as normal or attack. In addition, these datasets provide four categories of attacks: Denial-of-service (DOS), Probe, remote-to-local (R2L) and user-to-root (U2R). Many have criticized the KDD'99 dataset [48].

The DARPA 1999 Dataset is the widest and most comprehensive compared with its counterparts. This dataset has been compiled by MIT University Lincoln laboratory to assess intrusion detection systems; the DARPA contributed to the funding used for this research [49]. This database provided many new types of attacks for all four categories: probe, (U2R), (Dos) and (R2L), which included more than 200 intrusion instances of 58 different types of attacks, all of which were captured from the traffic. Moreover, this dataset presented new attack types which had not been presented either in the 1998 DARPA or KDD'99. These datasets have been evaluated by more than 80 different IDSs during the experiment period.

The DARPA 99 consists of five weeks, each of which includes five days. These weeks are classified into two types. The first type contains the first and third weeks; in fact, these two weeks represent a free attack, which can be used as training data. The second type, which contains the second, fourth and fifth weeks, is used to evaluate the intrusion-detection systems. Below are some of the reasons why the DARPA 99 is selected to test and evaluate the proposed system:

a) Volume of dataset: the size of the dataset totals more than 22GB and spans 12 hours per day, for 5 weeks, with each week consisting of 5 days.
b) Planned environment: data is captured by simulating the network traffic instead of being captured from a live network; the existence of protocols and events is intentional. Furthermore, these datasets comprise two weeks, with no intrusion event; the datasets are very interesting when it comes to determining the performance of false alarms for network intrusion systems. During the second, fourth and fifth weeks, specific attacks have been introduced in certain instances.
c) Comparability: there exist many published studies which have used the DARPA 1999 dataset. This means that it is possible to compare the results of techniques which have been developed in the systems proposed by past studies.

2.3 Related work

In this part of the thesis, we present most of the previous work related to the main goal of this thesis. The present section puts forth an overview of all research studies which have used the Alert Reduction techniques; such techniques reduce the quantity of these alerts to improve the quality of the networks security. The main objective of these techniques is to reduce duplicate and redundant alerts. Duplicate alerts are alerts that are triggered by a single event or triggered when more than one IDS generate alerts for the same network packets [57]. Redundant alerts are typically irrelevant alerts, and also represent false positive alerts, since these alerts refer to an alert which is either triggered by a normal traffic network event, or by the misuse configurations of the IDSs. With most real IDSs, the proportion of redundant and duplicate alerts is very high, standing at around 99% [3].

2.3.1 Alert Reduction Techniques

Several researchers have proposed new methodologies and designed systems in order to minimize the rate of false alarms and make it possible to differentiate between false positive alerts and real alerts. Such researchers include Hachmi and Limam (2013), Mohiuddin and A. (2014) [72], Alharby and Imai (2005) [51], Autrel and Cuppens (2005) [52], Jan et al. (2009) [53], and Viinikka et al. (2009) [54]. These researchers have learned about and analyzed the alerts generated by IDSs. Each alert contains a number of attributes. The above-mentioned researchers have studied all alerts, as well as their attributes; indeed, these attributes have been taken into account and analyzed closely. This analysis gives the researchers the ability to discover and propose effective methods which have already been proven to reduce the number of false alarms. As such, this section aims to cover these approaches and techniques, all of which are used to reduce the number of false positives alerts.

Mohiuddin and A. 2014 [72] proposed a new method based on improving the K-Means algorithm. Their approach is called CAD (Collective-Anomaly-Detection), while their improved algorithm is known as X-Means. They employed this improved algorithm to cluster the alerts and thus distinguish the anomaly alerts from normal alerts by measuring the similarity between the alerts dataset. Following this, they clustered the anomaly alerts to detect the Dos attacks. Their approach is examined by using DARPA datasets, and based on only four attributes, including Des_IP, Src_IP, Protocol_type, and Payload_length to detect DoS attacks. In actual fact, their approach focused on detecting only the Dos attacks, despite the fact that these alerts contain other types of attacks, such as U2R, R2L, and Probe.

In another alert reduction hybrid method, Hachmi and Limam (2013) designed a hybrid system to solve the problem of IDS alerts; their system consists of two stages: filtering stage and alarm correlation stage. They employed several algorithms in these two stages; for the first stage they used the SOM neural network algorithm and the K-Means algorithm, while in the second stage they used the neural GAS algorithm with FCM. To reduce the false positive rate and remove duplicated alerts from the intrusion detection system logs file, they employed the SOM neural network and K-Means algorithms in the first stage. The aim of this was to classify the alerts and remove the redundant generated alerts by looking for similarities and correlations between alerts based on the similarities of the features selected from the datasets; they extracted a total of three attributes (time stamp, source and destination IP addresses). They used the DARPA 99 Dataset and selected certain days from this dataset in order to improve their proposed system.

These authors applied the SOM algorithm in order to produce the associated map; the result of this algorithm classifier is a set of neurons, which are treated as the input for the second K-Means algorithm. This input contains a certain number of clusters, each of which is grouped into four similar neurons; together, these groups are triggered by single traffic at a specific time. In the second stage, the authors used the GAS and FCM algorithms to reduce the number of false positive alerts; in order to achieve this, it is first vital to identify the false alerts, following which the alerts rate can be reduced. In the second stage, they utilized a set of clusters which was generated by these two algorithms, and produced a binary classification based on certain features extracted from the dataset [59].

Another approach, this time based on data mining, was proposed by Zhang and Al-Mamory (2010). The main objective of this technique is to reduce the number of false positive alerts. This approach was designed in order to create a generalized alert for each cluster when the alerts are gathered into sets of clustering by this approach. To reduce the number of future alerts, this technique converts the root causes into a filter. With this proposed technique, the concept of nearest neighboring and generalization is taken into account.

This technique uses a new measurement to calculate the distances among alert feature values by performing a calculation process based on the background knowledge of the observed network. The results show that the reduction average stands at around 82% of the total alerts which are used [60].

Along similar lines, Alharby and Imai (2005) proposed a new method for reducing the number of false positive alerts generated by the IDSs. Their method aims to provide accurate information which helps to clarify whether or not an alert is normal. The objective of this method is accomplished by using continuous and discontinuous patterns. This approach flags any sets of sequence alerts when it is impossible to classify whether or not these alerts indicate abnormal activity.

A systematic model was constructed and used to ease some of the restrictions. The aim of said model is to understand the future alerts, and thus the proposed approach was based on ensuring the use of the all previously-collected alert patterns by using the sequential pattern extraction. The newly-extracted sequence pattern was similar to the proposed system, and exhibited a historical sequential pattern that had been extracted before. According to this proposed system, the extracted pattern represented the normal activity. Therefore, the probability of having normal activity is very high when the proposed system has several similarities with the pattern which was extracted.

This proposed system had flaws, the most important of which was that the approach focused on the size of the window of time for each alert set, depending on the accuracy classification. Indeed, if there is any miss in the time threshold, this would absolutely lead to the extraction of different alert patterns based on the size of the time window. In addition, similarities found by this system were based on the abnormal alerts pattern.

This proposed system from Alharby and Imai (2005) is evaluated through the use of the DARPA 1999 dataset, which uses only the first week, second week and third week. Furthermore, they collected their own dataset by conducting Snort to generate data for two weeks, following which they compared the results between these two datasets [51].

Yet another new system was proposed by Nien-yi Jan et al. (2009). The new system proposed by these authors was designed to monitor the network behavior on-line. This proposed system depends on a decision support system by building the classification of the alerts pattern behavior for this network behavior. The researchers in this system work depending on an off-line analysis. They constructed a decision support system which comprises three phases: the rule refining phase, the model constructing phase, and the alert preprocessing phase. The alert classification procedure model consists of three types of classification rule classes for alerts, including: 1) intrusion behavior classification rule class, 2) normal behavior classification rule class, and 3) suspicious behavior classification rule class. These phases are used to display and analyze each alert, while also flagging said alerts.

Based on the conducted experiment, this proposed system has shown good results and proved to be effective. Moreover, the proposed system is able to run on-line monitoring, while the system also enables domain experts to quickly and accurately discover suspicious behavior patterns, eases the workload of on-line alert analysis for the administrators, and effectively reduces the number of false alerts by up to 80%. However, the system also suffers with some common limitations.

Firstly, in order to ensure that the alert sequences are properly flagged, the classification rules must be frequently refined. This requires high computational overheads and sufficient domain knowledge from the experts. In fact, labeled data (rules) is not readily available in most cases. With a very large volume of network data encountered, it is certainly expensive to classify said data manually.

Secondly, since the system can only support a limited number of rules (up to 200 rules) in each classification class, said rules must be wisely selected in order to provide adequate coverage for all attack variants. Finally, the system does not appear to be cost efficient enough, since the rule classes are created for each target host (each sensor). A significant number of rules will be required for IDSs implemented in a multi-host network environment [63].

Viinikka et al. (2009) presented a novel system that aggregates alerts into an alert sequence. Two basic assumptions were applied in this study: first, normal system behaviors in an alert flow can be identified by looking for regularities and smooth changes in the alert intensity. Second, the normal behavior is not observable for an individual alert, but is observable with an alert sequence (flow). To follow these hypotheses, the system modeled the regularities of the alert flows from the normal behaviors and used the created model to filter out the irrelevant or low impact alerts from the alert log.

Although the system exhibited great performance, there remains a risk when modeling abnormal behaviors into a normal behavior model; this risk emerges if there has been no detection of abrupt changes in the alert intensity caused by true alerts [54].

Alshammari et al. (2007) [11] built a hybrid neuro-fuzzy system to reduce the number of false positive alerts, and took into account the effect of adding background knowledge to the alerts data. This system is based on several factors, including Membership Function (MF), Fuzzy Sets Variables, Rule Weights, Learning rate, Cross-validation, and Number of epochs. The MFs provided by the NEF-CLASS include Trapezoidal, Triangular, List function, and Bell-shaped. In actual fact, they used all four of these MFs with Confidence Factor (CF=0.99) as well as background knowledge classes. Moreover, they used different Fuzzy variables (3, 4, 5). This system is tested by using the first three weeks of the DARPA 1999 dataset, while the remaining weeks (the fourth and fifth weeks) are used to evaluate the system. Their system efficiently reduces the number of false alarms by 90.92% through the use of an IP class with 10 features, 4 variables, and a fuzzy set with trapezoid MF.

CHAPTER 3

Methodology and Implementation of the proposed system

3.1 Introduction

This chapter explains the method and the architecture used for the proposed system and aims to describe the implementation of said system. The main goal of designing this system is to remove the duplicated alerts and evaluate the redundant alerts by calculating the degree of threat for each alert. This will subsequently minimize the number of false positive alerts from intrusion-detection systems, thus leading to high quality security for the networks.

The widespread utilization of the internet and networks in various specialties and fields has increased. Although these technologies have led to an evolutionary revolution in all fields of life, said technologies still suffer from some security weaknesses, some of which are now becoming more of a concern. The main reason for this development is linked to the malicious programs which continue to develop and progress; indeed, this eventually has a negative impact on the security quality of the networks. These problems have encouraged most researchers to propose new approaches aimed at improving the networks security so as to overcome said issues.

As mentioned in Chapter 2, Section 2.3, scholars have proposed many approaches aimed at developing intrusion-detection systems and improving the protection of data transmission over the internet and networks. In actual fact, the IDS alerts are presented to the security analyst, who will check said alerts based on several rules and gauges. Indeed, most of these alerts are false alarms (FP), which complicate the functions of the system for the security analyst, and prevent him/her from completing his/her duties as required. In this thesis, we propose a new system to reduce and evaluate the alerts in the IDSs' logs file. The proposed system is designed by merging two phases: the Filtering Duplicated Alerts Phase, and the Alerts Threat Evaluation Phase.

3.2 The Proposed System Architecture

In order to gain an overall view of all the components of the proposed system in this section, we will explain the two phases of this system in detail: the Filtering Duplicated Phase, and the Alerts Threat Evaluation Phase.

Furthermore, both of these phases are implemented in order to construct a single system so as to enhance the performance of the proposed system; this will also lead to higher efficiency and more accurate results. These two phases will be discussed more rigorously in the implemented experiment and discussion results in the next chapters.

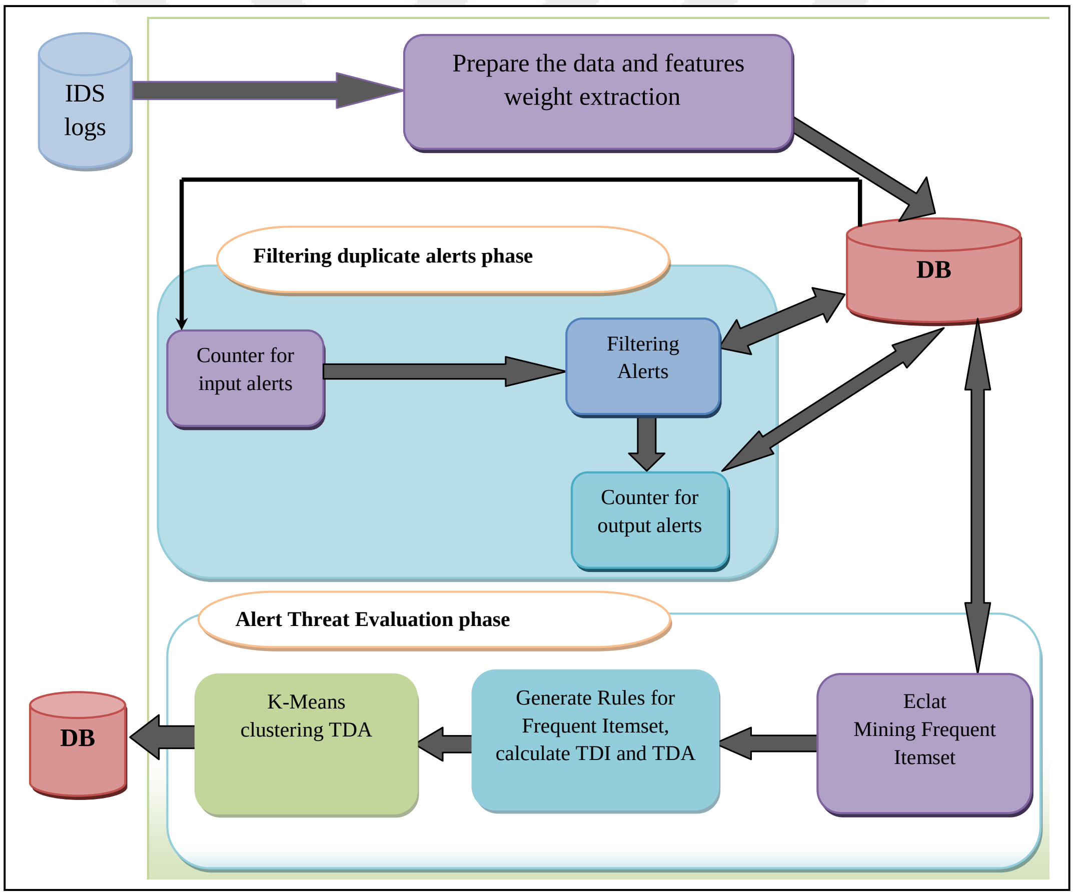

IDS logs
Prepare the data and features weight extraction
DB
Filtering duplicate alerts phase
Counter for input alerts
Filtering Alerts
Counter for output alerts
Alert Threat Evaluation phase
DB
K-Means clustering TDA
Generate Rules for Frequent Itemset, calculate TDI and TDA
Eclat Mining Frequent Itemset

Figure 3.1 Architecture of the Proposed System

These two phases cannot be utilized independently, as using one of these phases alone is not sufficient; indeed, it would not be workable to use the Filtering

Duplicated Alerts Phase alone without the Alerts Threat Evaluation Phase. The result of this phase is not sufficient for the systems analyst, and so these results still generate a large number of FPs. Such is the scale of the FPs that the systems analyst does have the capabilities or scalability to differentiate between the false alarms and the real alarms. In actual fact, this phase is designed and developed in order to overcome the problems which have been addressed in detail in Chapter 2. Indeed, as outlined in Chapter 2, a single attack causes the IDSs to generate hundreds of alerts due to reasons which have already been mentioned above.

Each phase of this proposed system is responsible for running its functionality. The proposed program is executed through the implementation of the following steps:

1. Receive the alerts data as an Excel file.

2. Based on the results achieved by AlSaedi et al. (2012) [73], select the highest weight attributes.

3. Use a counter to count the input alerts.

4. Use call filtering duplicated alerts to:
   a. Filter alerts based on the similarities between the attribute items: (If $Alert_i$ [A_1, A_2…, A_n] = $Alert_j$ [A_1, A_2…, A_n] Then Remove $Alert_j$ from Alerts_file).

   b. Filter alerts based on the similarities between the attribute items with time attributes: (If $Alert_i$ [A_{Time}] <= $Alert_j$ [A_{Time}] ,And $Alert_i$ [A_1, A_2, A_3…, A_n] = $Alert_j$ [A_1, A_2, A_3…, A_n] Then Remove $Alert_j$ from Alerts_file).

5. Call the Eclat algorithm in order to mine frequent itemsets.

6. Call the Generate Rules algorithms in order to:

a. Generate Rules of Frequent Itemsets
b. Calculate threat degree for each item:

$$\mathrm{TDI} = \frac{\sum_{i=1}^{s} \mathrm{conf}_I}{s} \tag{3.1}$$

c. Calculate threat degree for each alert:

$$\mathrm{TDA} = (\sum_{j=1,k=0}^{\mathrm{No\_Attributes}} (\mathrm{TDI}))/\mathrm{No\_Selected\_\,Attributes} \tag{3.2}$$

7. Call K-Mean to cluster the TDA values.

3.3 Prepare the data and Attributes extraction

This step is used at the beginning, and only once. The aim of this process is to identify the attributes of the alert which are most effective; the process of the system involves extracting the standard attributes from the alerts database. This system provides the system analyst and system administrator with the ability to choose the significant attributes. Finally, the implementation of the proposed system is based on the seven most effective attributes; this makes it possible to achieve the best results with the highest efficiency.

At this point it is important to refer to the results achieved by AlSaedi et al. (2012) [73], who applied the Gain Info algorithm to extract the weight of each attribute for all five weeks of the DARPA 99 dataset; indeed, those results provided us with a selection of attributes with the highest weight, which in turn gave us the desired results from the proposed system design. In addition, their findings mean that we have the ability to select the best attributes; this is clearly shown in Table 4.1, which illustrates the results of the Information Gain Ratio algorithm which were generated by [73]:

Table 3.1 Results of the Information Gain Ratio algorithm generated by [73]

Feature name	Weighted
Ips	2.0759
Time	2.0712
Ports	1.9529
Portd	1.9038
Ipd	1.4476
Priority	1.2943
RFC	1.0797
Protocol	0.7858
Date	0.5197
TTL	0.3167
Code	0.2451
Type	0.2103
G:S:RID	0.1518
DgmLen	0.1443
IpLen	0.1003
ID	0.0441

3.4 Filtering Duplicate Alerts (FDA) Phase

The FDA Phase is the first step in the proposed system, and it is certainly worth deleting the duplicated alerts, as this results in minimizing the number of false alerts. More specifically, this phase has three sub-phases, including: counter for input alerts, filtering alerts, and counter for output alerts from this phase. The architecture of this phase is illustrated in Figure 3.2 and Figure 3.3, which show the main content and provide further details on the FDA Phase.

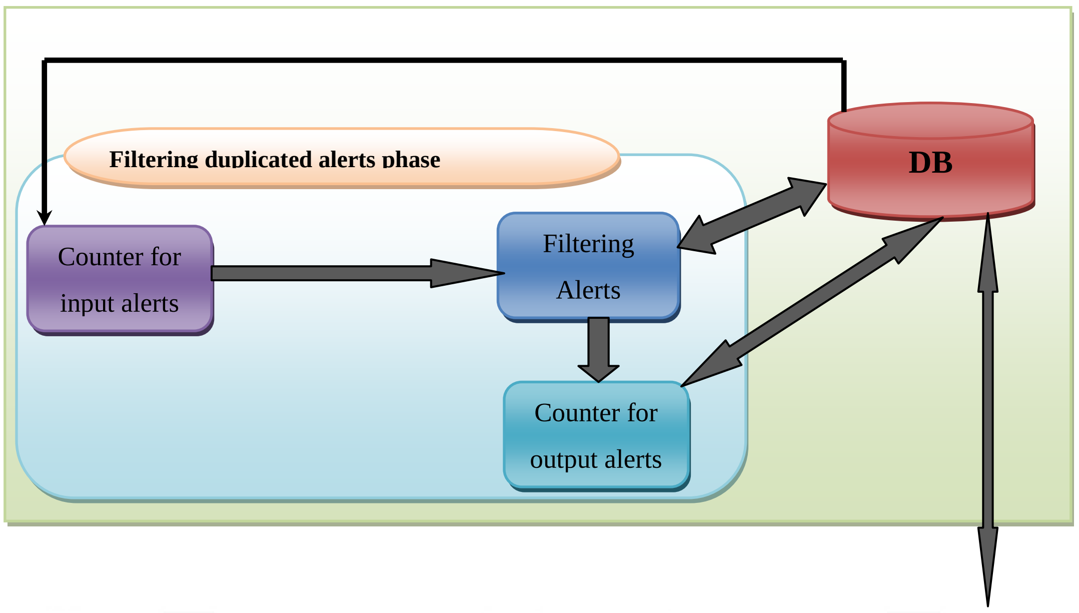


Figure 3.2 FDA Phase Architecture

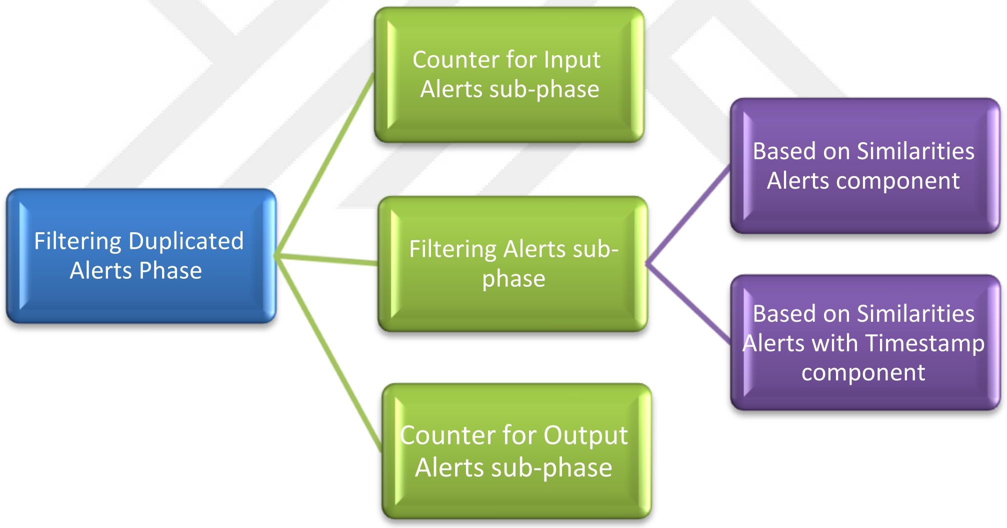

Filtering Duplicated Alerts Phase
Counter for Input Alerts sub-phase
Filtering Alerts sub-phase
Counter for Output Alerts sub-phase
Based on Similarities Alerts component
Based on Similarities Alerts with Timestamp component

Figure 3.3 Contents of the FDA Phase

3.4.1 Counter for Input Alerts sub-phase

The main objective of this sub-phase is to count the total number of the alerts that are inputted into the FDA Phase and to check the available components of those alerts. This sub-phase process is carried out before the Filtering Alerts sub-phase process is implemented.

3.4.2 Filtering Alerts sub-phase

The main goals of the Filtering Alerts sub-phase are to decrease the number of alerts by eliminating the duplicated alerts, which reduces the number of FPs [64]. The Filtering Alerts sub-phase components constitute the core part of the Filtering Duplicated Alerts Phase. The significant functions of this sub-phase are to eliminate duplicated alerts based on both factors, identify the similarities between the attributes of the alerts, and identify the similarities between the attributes of the alerts with timestamp attributes. The Filtering Alerts sub-phase comprises two components: (1) Based on Similarities Alerts component; and (2) Based on Similarities Alerts with Timestamp component. This is demonstrated in Figure 3.4.

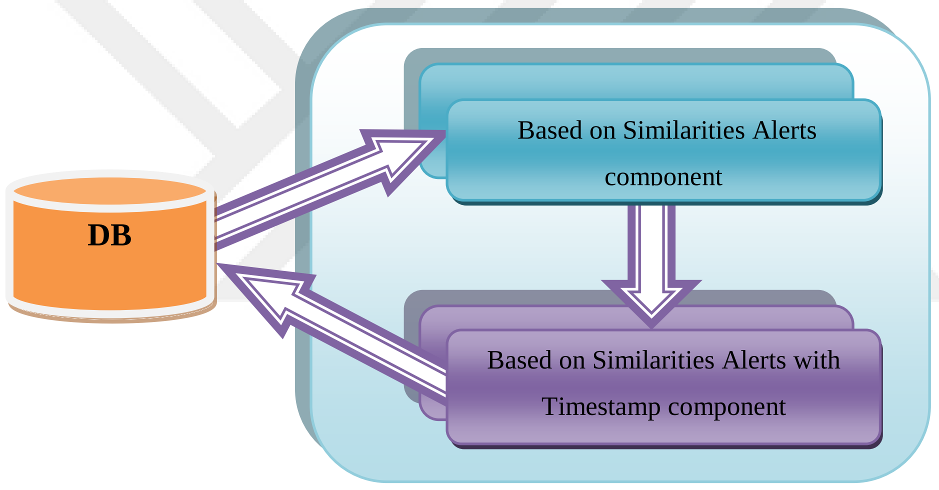
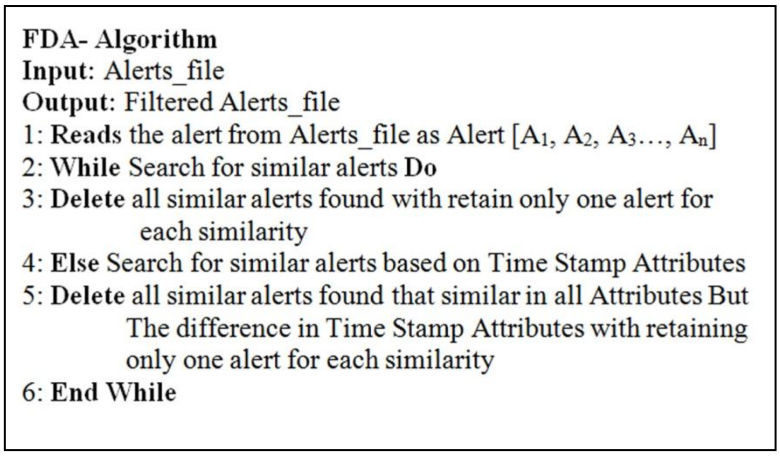

DB
Based on Similarities Alerts component
Based on Similarities Alerts with Timestamp component

Figure 3.4 Components of the Filtering Alerts sub-phase

The FDA phase aims to remove duplicated alerts that represent the main body of false positive alerts. This is achieved by proposing a method to filter these alerts. In the present section, we explain the implementation of this method. Figure 3.5 displays the pseudocode of the FDA phase.



```
FDA- Algorithm
Input: Alerts_file
Output: Filtered Alerts_file
1: Reads the alert from Alerts_file as Alert [$A_1$, $A_2$, $A_3$…, $A_n$]
2: While Search for similar alerts Do
3: Delete all similar alerts found with retain only one alert for
        each similarity
4: Else Search for similar alerts based on Time Stamp Attributes
5: Delete all similar alerts found that similar in all Attributes But
        The difference in Time Stamp Attributes with retaining
        only one alert for each similarity
6: End While
```

Figure 3.5 Pesudocode of FDA phase

As shown in the above figure, this phase passes through two components. The first is based on the similarities between items of alert attributes. There are two pointers which are tasked with reading the items of the alerts; indeed, the first alert from the database is read by the first pointer, while the second alert is read by the second pointer, so that it is possible to make a comparison between the items of alerts attributes for these two alerts. The second alert will be deleted, while the first alert will only be kept if the similarity exists; if there exists no similarity, the pointers move from the second alert to the third alert, in order to re-compare another two alerts, and so on.

The second component is based on similarities between alerts and Timestamp attributes. This component is used to remove the duplicated alerts when similarities are found between items of the alert attributes; the only differences are found in the timestamp, and these timestamp values are very close to each other, thus indicating that these alerts are generated from a single event in successive and very close time periods. For instance, Table 3.2 represents the given alerts and is a part of the database alerts:

Table 3.2 Part of Alerts Database

A-ID	G:S:RID	RuleFileComes	Priority	Date	TimeHM	TimeSMs	Ips	Ports	Ipd	Portd
1	1:1201:8	ATTACK-RESPONSES 403 Forbidden	2	8-Mar	21:01	13.339084	172.16.112.100	P80	206.48.44.18	P1059
2	1:1201:8	ATTACK-RESPONSES 403 Forbidden	2	8-Mar	21:01	13.339084	172.16.112.100	P80	206.48.44.18	P1060
3	1:1201:8	ATTACK-RESPONSES 403 Forbidden	2	8-Mar	21:01	13.346245	172.16.112.100	P80	206.48.44.18	P1061
4	1:1201:8	ATTACK-RESPONSES 403 Forbidden	2	8-Mar	21:01	13.350848	172.16.112.100	P80	206.48.44.18	P1062
5	1:384:5	ICMP PING	3	8-Mar	21:06	44.388716	192.168.1.2	-p	192.168.1.1	-p

In order to implement the Filtering duplicated alerts phase for the above data, this data should pass through two components. The first component looks for similarities between alerts based on similarities between attributes of the items. Alert-1 is selected, following which alert-2 is read in order to make a comparison between the items of the attributes. If an obvious similarity is found, this component will eliminate alert-2 and retain alert-1. With regard to the other alerts, no similarities are found among them. The results of this component are explained in Table 3.3 below.

Table 3.3 Results of First Component Process

A-ID	G:S:RID	RuleFileComes	Priority	Date	TimeHM	TimeSMs	Ips	Ports	Ipd	Portd
1	1:1201:8	ATTACK-RESPONSES 403 Forbidden	2	8-Mar	21:01	13.339084	172.16.112.100	P80	206.48.44.18	P1059
3	1:1201:8	ATTACK-RESPONSES 403 Forbidden	2	8-Mar	21:01	13.346245	172.16.112.100	P80	206.48.44.18	P1061
4	1:1201:8	ATTACK-RESPONSES 403 Forbidden	2	8-Mar	21:01	13.350848	172.16.112.100	P80	206.48.44.18	P1062
5	1:384:5	ICMP PING	3	8-Mar	21:06	44.388716	192.168.1.2	-p	192.168.1.1	-p

The second component is accountable for the similarity alerts with the timestamp attributes. This essentially means that when alert-1 is selected and alert-3 is read, a similarity has been found among all items of the alerts attributes, but the only difference is in the TimeSMs attribute. As such, the component will check if this alert is the same in both the Date and TimeHM attributes, following which alert-3 will be deleted and classed as a duplicated alert which is generated from that single event; only alert-1 will be retained. This process is identical for alert-4, which has also been deleted, while alert-1 has been retained. Finally, no similarities are found among the items of the alerts attribute, and so the obtained results for this component will be elaborated on in Table 3.4

Table 3.4 Results of Second Component Process

A-ID	G:S:RID	RuleFileComes	Priority	Date	TimeHM	TimeSMs	Ips	Ports	Ipd	Portd
1	1:1201:8	ATTACK-RESPONSES 403 Forbidden	2	8-Mar	21:01	13.339084	172.16.112.100	P80	206.48.44.18	P1059
5	1:384:5	ICMP PING	3	8-Mar	21:06	44.388716	192.168.1.2	-p	192.168.1.1	-p

3.4.3 Counter for Output Alerts sub-phase

This sub-phase is the final process of the first phase (Filtering Duplicated Alerts Phase) and is responsible for countering the remaining alerts which result from the Filtering Alerts sub-phase. The output from this phase is the input for the Alert threat evaluation phase.

3.5 Alert Threat Evaluation phase (ATE)

The ATE phase is the second phase, and is considered the heart of the methodology of the proposed system. The purpose of this phase is to calculate the threat degree values for each alert by generating frequent itemsets and producing the rules for those itemsets with the threat degree values for each item; finally, this phase also clusters the TDA values. Figure 3.6 shows the architecture of the ATE Phase.

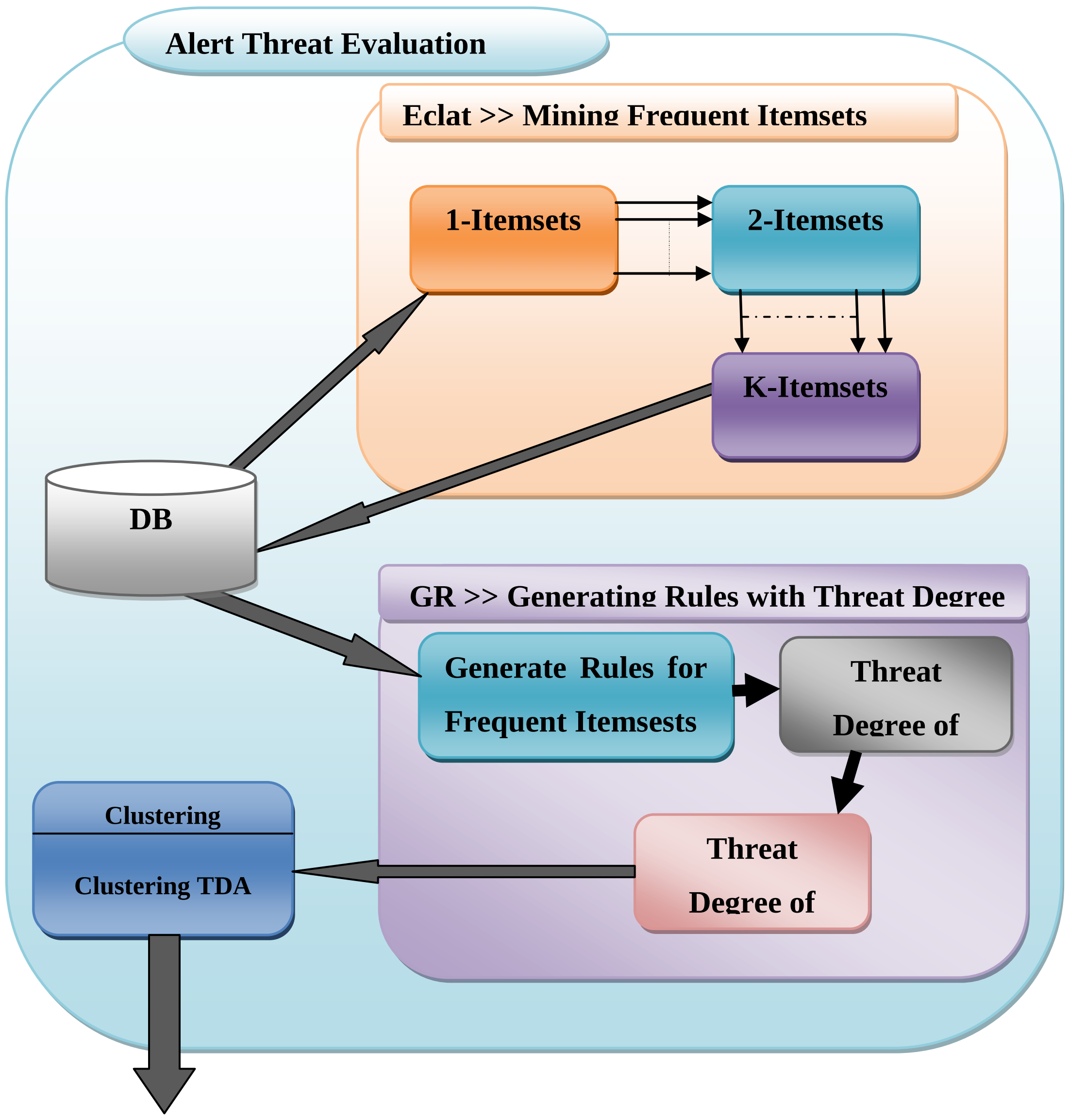

Alert Threat Evaluation
Eclat >> Mining Frequent Itemsets
1-Itemsets
2-Itemsets
K-Itemsets
DB
GR >> Generating Rules with Threat Degree
Generate Rules for Frequent Itemsests
Threat Degree of
Threat Degree of
Clustering
Clustering TDA

Figure 3.6 Architecture of the ATE phase

The ATE phase consists of three algorithms: the Eclat algorithm, the Generate Rules algorithm, and the Clustering algorithm. These algorithms have been improved so as they are more efficient and easy to implement on items of the alert attributes; said algorithms are also more appropriate when it comes to calculating the threat degree for each item for the alert attributes and the evaluation of these alerts. Figure 3.7 presents a more detailed explanation of the content of the Alert Treat Evaluation phase.

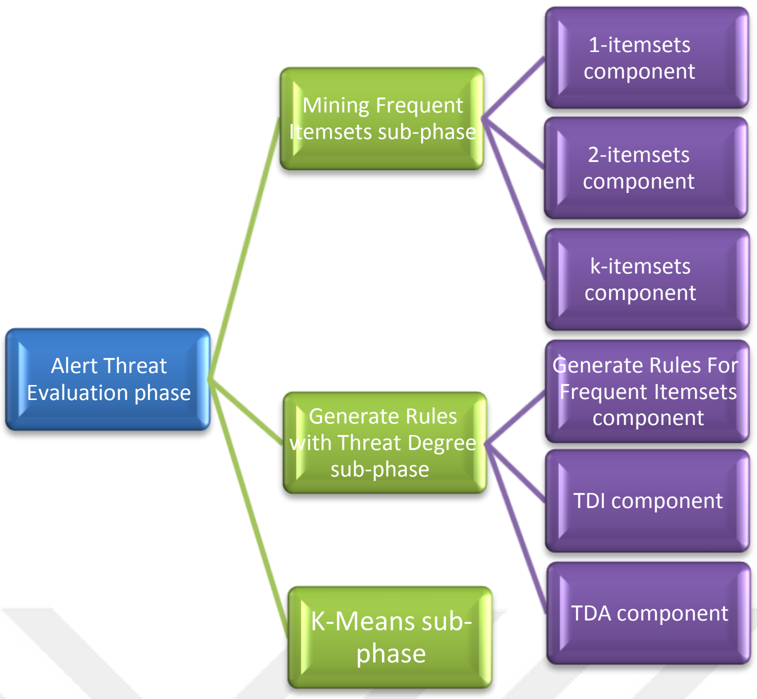

Alert Threat Evaluation phase
Mining Frequent Itemsets sub-phase
Generate Rules with Threat Degree sub-phase
K-Means sub-phase
1-itemsets component
2-itemsets component
k-itemsets component
Generate Rules For Frequent Itemsets component
TDI component
TDA component

Figure 3.7 Contents of ATE Phase

3.5.1 Mining Frequent Itemsets sub-phase

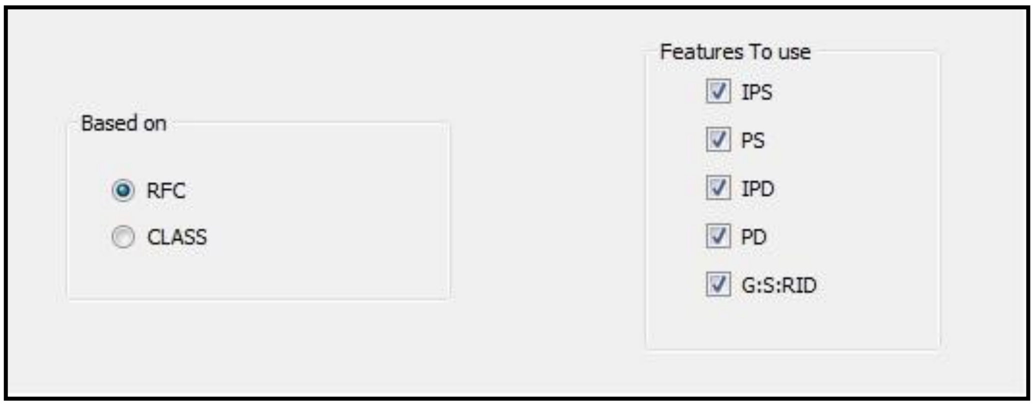
This sub-phase is the first process of the second phase of the proposed system. The Mining Frequent Itemsets sub-phase is accomplished by using the Eclat algorithm to generate an itemset with its Tidlist by implementing the Union between itemsets and the Intersection between the Tidlist items. The support value of the frequent itemset is calculated by measuring the length of its Tidlist.

The Eclat algorithm is enhanced to generate frequent itemsets from each single attribute of the selected attributes of the database. Moreover, this algorithm is able to find the relation between these frequent items in the single attribute. Furthermore, the algorithm is more flexible to different data types within the database, such as binary data, text data, numerical data, and so on.

The Eclat algorithm needs to scan the database only once; there is no need for this algorithm to rescan the database several times in order to generate the second and third itemsets, and so on. This is why the Eclat algorithm is faster than the Apriori and

Fp-Growth algorithms. After the algorithm generates all of the frequent 1-itemsets, the algorithm generates 2-itemsets entirely from 1-itemsets without any need to rescan the database. This process continues, and proceeds to generate k-itemsets based on k-1 itemsets until no frequent itemsets are found in the previous frequent itemsets. In actual fact, the Mining Frequent Itemsets sub-phase consists of three components, namely 1-itemsets, 2-itemsets and k-itemsets.

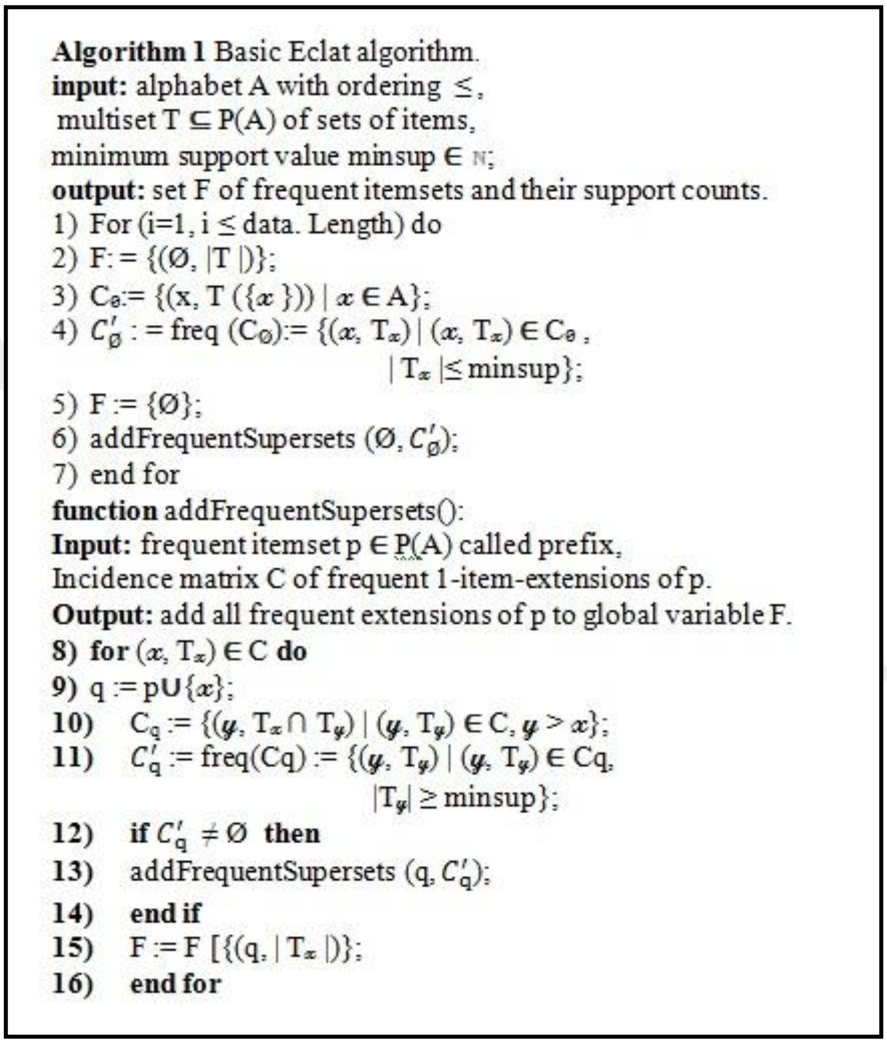
To assist this algorithm in working well, it is necessary to convert the database from horizontal data format into vertical data format. The proposed system provides the flexibility by presenting two options which can act as the Tidlist for frequent items, including: RuleFileComes or Classification attributes. However, the experiments in this thesis are based on selecting the RuleFileComes attribute as the Tidlist, and selecting the IPS, PS, IPD, PD and G:S:RID attributes to mine their frequent itemsets. In fact, the RuleFileComes attribute of alerts data has been considered as the basis for triggering the alerts which are generated, and represents the rules that the IDSs use to generate alerts. Figure 4.3 depicts the choices that are provided by the proposed system to users, such as systems analysts or systems administrators.


Based on
[✓ RFC] CLASS

Features To use
✓ IPS
✓ PS
✓ IPD
✓ PD
✓ G:S:RID

Figure 3.8 Options of the Proposed System of the attributes that could act as Tidlist or to mine their items.

This algorithm performs its process on the alerts data which results from the first phase of the proposed system. Depending on the attributes selected by the system user, this improved algorithm will divide the database into several parts to perform its functions on these parts while maintaining the original data order. Each part is a vertical database consisting of two columns, including: Items and Tidlist. As previously mentioned, the Eclat algorithm scans the alerts database just once. Figure 3.9 elaborates on the pesudocode of the Eclat algorithm [74].



```
Algorithm 1 Basic Eclat algorithm.
input: alphabet A with ordering $\leq$,
 multiset $T \subseteq P(A)$ of sets of items,
minimum support value minsup $\in \mathbb{N}$;
output: set F of frequent itemsets and their support counts.
1) For (i=1, i $\leq$ data. Length) do
2) F: = $\{(\emptyset, |T|)\}$;
3) $C_\emptyset := \{(x, T(\{x\})) \mid x \in A\}$;
4) $C'_\emptyset$ : = freq $(C_\emptyset) := \{(x, T_x) \mid (x, T_x) \in C_\emptyset$ ,
                                  $|T_x| \leq$ minsup$\}$;
5) F := $\{\emptyset\}$;
6) addFrequentSupersets $(\emptyset, C'_\emptyset)$;
7) end for
function addFrequentSupersets():
Input: frequent itemset $p \in P(A)$ called prefix,
Incidence matrix C of frequent 1-item-extensions of p.
Output: add all frequent extensions of p to global variable F.
8) for $(x, T_x) \in C$ do
9) q := $p \cup \{x\}$;
10)     $C_q := \{(y, T_x \cap T_y) \mid (y, T_y) \in C, y > x\}$;
11)     $C'_q$ := freq(Cq) := $\{(y, T_y) \mid (y, T_y) \in Cq$,
                                  $|T_y| \geq$ minsup$\}$;
12)     if $C'_q \neq \emptyset$ then
13)     addFrequentSupersets (q, $C'_q$);
14)     end if
15)     F := F $[\{(q, |T_x|)\}$;
16)     end for
```

Figure 3.9 Pesudocode of the Eclat Algorithm

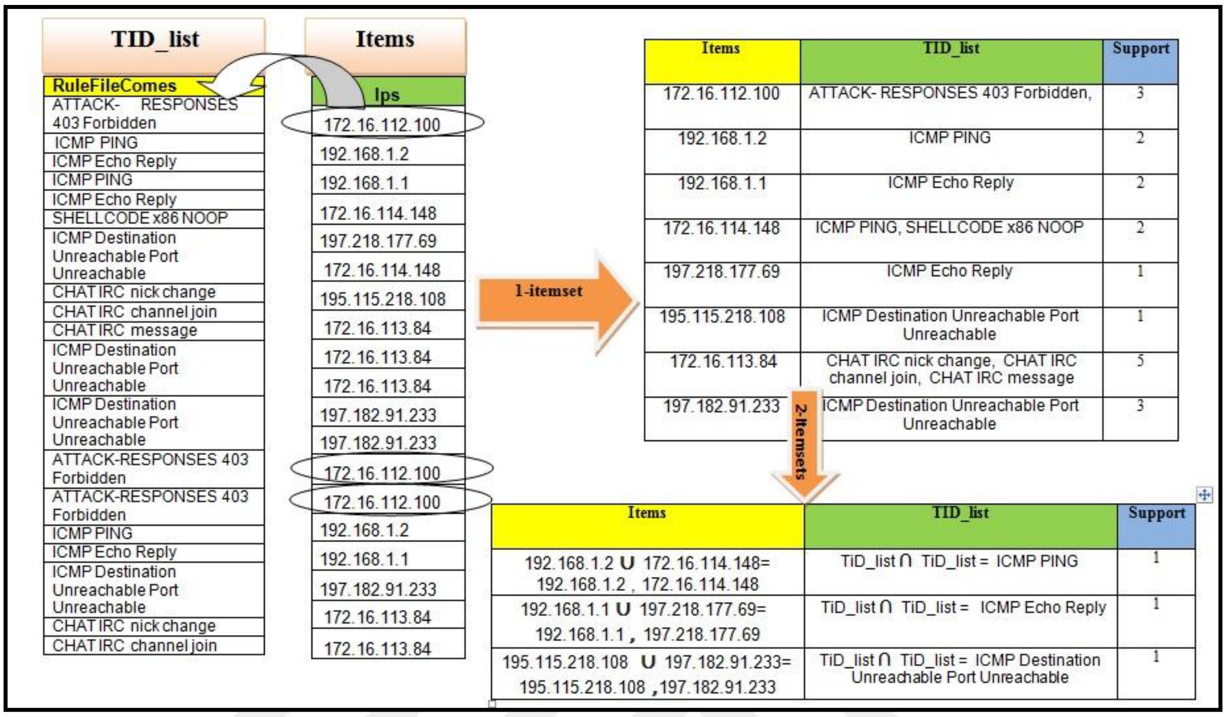
This algorithm has been explained in detail by Schmidt-Thieme (2004) [74]. The processes of the algorithm are as follows: firstly, the algorithm loads the frequent 1-itemset, each item of which has its TID_list, and then generates all frequent itemsets from the 1-itemset without the need to scan the database again.

In the Eclat algorithm, as shown in Figure 3.9, step 3 groups the items ($C'_{\emptyset}$) with their Tidlist and measures the length of their Tidlist. Step 4 involves mining for frequent items whose Tidlist length is equal to or greater than the min_sup threshold from the candidate items which are generated in step 3. Following this, step 6 adds these frequent items into F, which represents the 1-itemset.

This function of the Eclat algorithm (from step 8 to step 16) now scans only the frequent items that have been generated from the previous process, without the need to rescan the original database. To generate the 2-candidate frequent itemsets (C_q) from F, which is called the prefix, it is essential to join each pair of items and then find the intersection in their Tidlists; this process is carried out in step 10. Counting the length of the intersected Tidlists for each pair of items (C'_q) yields these values and their support. Step 11 involves searching for any pairs of items that have a support value which satisfies the min_sup threshold. Step 12 through to step 15 are used to add this generated 2-itemset in F to the process; this itemset again generates 3-itemsets. In actual fact, this process is repeated many times until no new frequent itemsets are found.

Indeed, this proposed system intentionally makes the min_sup threshold of the Eclat algorithm equal to 1. As such, there will be no infrequent items. Thus, the implementation of this algorithm in the present thesis will suffice, along with the findings for the 1-itemsets and 2-itemsets, due to the min_sup value which has been selected. The Eclat algorithm will generate many frequent items, because no infrequent itemsets are found; indeed, this is what prevents the algorithm from continuing to generate other sets of frequent itemsets.

For a more detailed insight, it is fitting to use the following example, which traces the behavior of the Eclat algorithm with min sup=1 based on [74]. Firstly, the algorithm selects 1-itemsets from database items, following which the algorithm joins each two items and intersects their Tidlists to generate the candidate 2-itemset (C'_q). This is shown below in Figure 3.10.



TID_list	Items
RuleFileComes	Ips
ATTACK- RESPONSES 403 Forbidden	172.16.112.100
ICMP PING	192.168.1.2
ICMP Echo Reply	192.168.1.1
ICMP PING	172.16.114.148
ICMP Echo Reply	197.218.177.69
SHELLCODE x86 NOOP	172.16.114.148
ICMP Destination Unreachable Port Unreachable	195.115.218.108
CHAT IRC nick change	172.16.113.84
CHAT IRC channel join	172.16.113.84
CHAT IRC message	172.16.113.84
ICMP Destination Unreachable Port Unreachable	197.182.91.233
ICMP Destination Unreachable Port Unreachable	197.182.91.233
ATTACK-RESPONSES 403 Forbidden	172.16.112.100
ATTACK-RESPONSES 403 Forbidden	172.16.112.100
ICMP PING	192.168.1.2
ICMP Echo Reply	192.168.1.1
ICMP Destination Unreachable Port Unreachable	197.182.91.233
CHAT IRC nick change	172.16.113.84
CHAT IRC channel join	172.16.113.84

Items	TID_list	Support
172.16.112.100	ATTACK- RESPONSES 403 Forbidden,	3
192.168.1.2	ICMP PING	2
192.168.1.1	ICMP Echo Reply	2
172.16.114.148	ICMP PING, SHELLCODE x86 NOOP	2
197.218.177.69	ICMP Echo Reply	1
195.115.218.108	ICMP Destination Unreachable Port Unreachable	1
172.16.113.84	CHAT IRC nick change, CHAT IRC channel join, CHAT IRC message	5
197.182.91.233	ICMP Destination Unreachable Port Unreachable	3

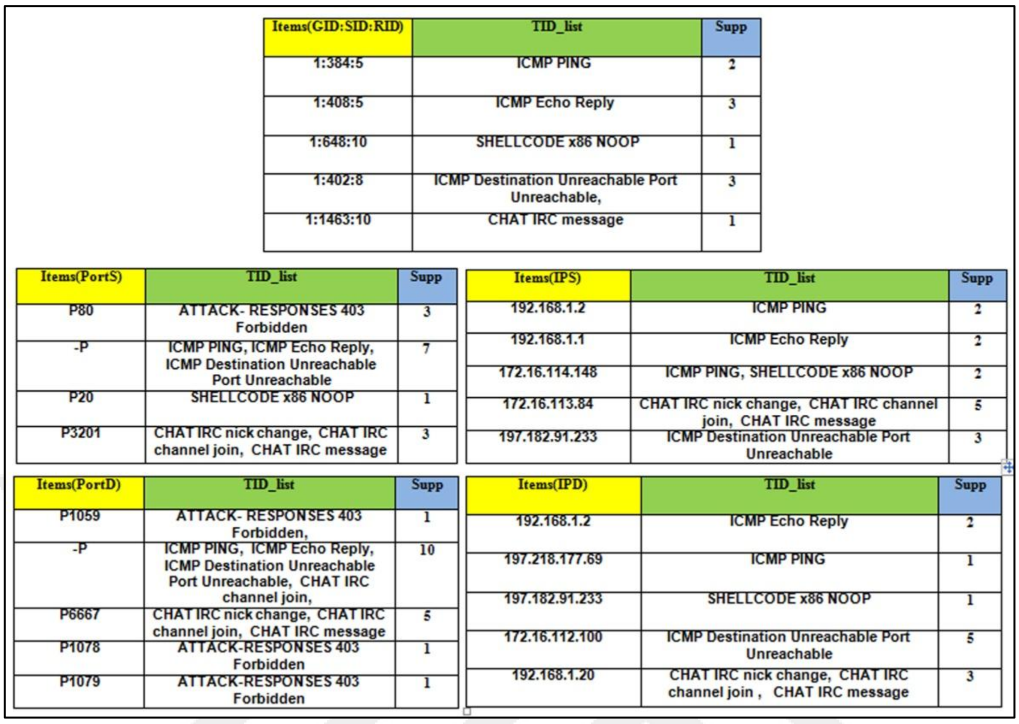
Items	TID_list	Support
192.168.1.2 **U** 172.16.114.148= 192.168.1.2 , 172.16.114.148	TiD_list ∩ TiD_list = ICMP PING	1
192.168.1.1 **U** 197.218.177.69= 192.168.1.1 **,** 197.218.177.69	TiD_list ∩ TiD_list = ICMP Echo Reply	1
195.115.218.108 **U** 197.182.91.233= 195.115.218.108 **,**197.182.91.233	TiD_list ∩ TiD_list = ICMP Destination Unreachable Port Unreachable	1

Figure 3.10 Implementation of Eclat Algorithm

The process of generating itemsets has passed through two components: the first of these is generate 1-iemsets, while the second is generate 2-itemsets.

3.5.1.1 Generation 1-itemsets

This component is performed depending on steps 1-5, which are shown in Figure 3.9. The first step is a loop which starts with the first items and ends with the last items of the data (For (i=1, i ≤ data. Length)). Step 3, which involves storing each item with their Tidlist, contains ($C_0 := \{(x, T(\{x\})) \mid x \in A\}$) and maintains the order of the original data. This thesis is based on 5-attribute items, including the IPS, PortS, IPD, PortD and GID:SID:RID attributes; there are also two attributes of Tidlist (RFC or Classification), as shown in Figure 3.11. In actual fact, the RFC attribute is selected in this experiment. Figure 3.11 explains the results after performing steps 1-4.



Items(GID:SID:RID)	TID_list	Supp
1:384:5	ICMP PING	2
1:408:5	ICMP Echo Reply	3
1:648:10	SHELLCODE x86 NOOP	1
1:402:8	ICMP Destination Unreachable Port Unreachable,	3
1:1463:10	CHAT IRC message	1

Items(PortS)	TID_list	Supp
P80	ATTACK- RESPONSES 403 Forbidden	3
-P	ICMP PING, ICMP Echo Reply, ICMP Destination Unreachable Port Unreachable	7
P20	SHELLCODE x86 NOOP	1
P3201	CHAT IRC nick change, CHAT IRC channel join, CHAT IRC message	3

Items(IPS)	TID_list	Supp
192.168.1.2	ICMP PING	2
192.168.1.1	ICMP Echo Reply	2
172.16.114.148	ICMP PING, SHELLCODE x86 NOOP	2
172.16.113.84	CHAT IRC nick change, CHAT IRC channel join, CHAT IRC message	5
197.182.91.233	ICMP Destination Unreachable Port Unreachable	3

Items(PortD)	TID_list	Supp
P1059	ATTACK- RESPONSES 403 Forbidden,	1
-P	ICMP PING, ICMP Echo Reply, ICMP Destination Unreachable Port Unreachable, CHAT IRC channel join,	10
P6667	CHAT IRC nick change, CHAT IRC channel join, CHAT IRC message	5
P1078	ATTACK-RESPONSES 403 Forbidden	1
P1079	ATTACK-RESPONSES 403 Forbidden	1

Items(IPD)	TID_list	Supp
192.168.1.2	ICMP Echo Reply	2
197.218.177.69	ICMP PING	1
197.182.91.233	SHELLCODE x86 NOOP	1
172.16.112.100	ICMP Destination Unreachable Port Unreachable	5
192.168.1.20	CHAT IRC nick change, CHAT IRC channel join , CHAT IRC message	3

Figure 3.11 Frequent 1-items generated for several attributes by executing first four steps of Eclat algorithm.

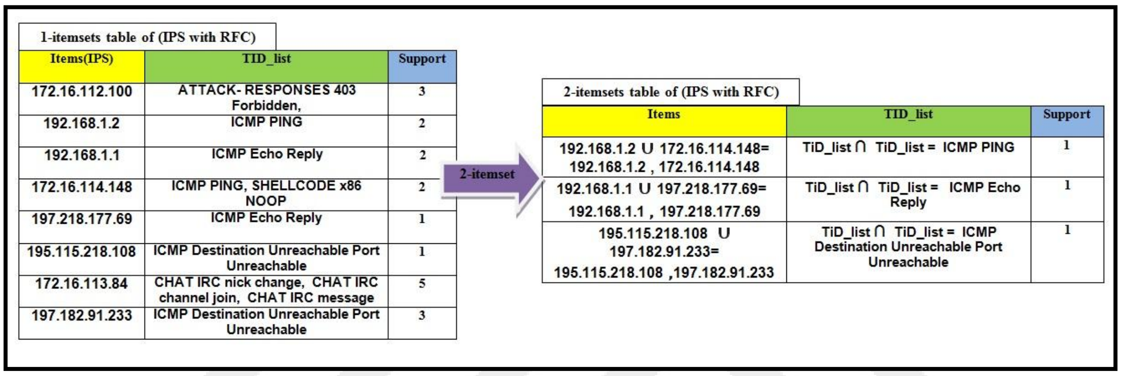
As shown in the above figure, each table is generated by applying the steps of the said algorithm, the aim of which is to generate 1-itemsets and represent the attribute items. Each table is stored as an Excel file.

This part of the algorithm generates a new database which consists of three columns: items, Tidlist, and support columns. Indeed, X represents items, while each item represents one alert attribute. The number of attributes and the Tidlists attribute are both selected by the user. This property provides system users with the flexibility to select what they see as the best option, while the proposed system is designed to be ready and work efficiently with different attributes that the user selects.

3.5.1.2 Generation 2-itemsets

In the Eclat algorithm, the iteration function (AddFrequentSupersets) steps from step 8 to step 15 are applied to generate 2-itemsets from 1-itemsets tables; these 2-itemsets are stored on the same file for each attribute. The input for this function is a table which represents 1-itemsets. The output from these steps consists of 2-itemsets.

Based on steps 8-15, with a min_sup threshold equal to 1, it is useful to recall the example in Figure 3.10. With this example, we can see a detailed explanation of how the 2-itemsets are obtained, as shown in Figure 3.12.


1-itemsets table of (IPS with RFC)

Items(IPS)	TID_list	Support
172.16.112.100	ATTACK- RESPONSES 403 Forbidden,	3
192.168.1.2	ICMP PING	2
192.168.1.1	ICMP Echo Reply	2
172.16.114.148	ICMP PING, SHELLCODE x86 NOOP	2
197.218.177.69	ICMP Echo Reply	1
195.115.218.108	ICMP Destination Unreachable Port Unreachable	1
172.16.113.84	CHAT IRC nick change, CHAT IRC channel join, CHAT IRC message	5
197.182.91.233	ICMP Destination Unreachable Port Unreachable	3

2-itemsets table of (IPS with RFC)

Items	TID_list	Support
192.168.1.2 ∪ 172.16.114.148= 192.168.1.2 , 172.16.114.148	TiD_list ∩ TiD_list = ICMP PING	1
192.168.1.1 ∪ 197.218.177.69= 192.168.1.1 , 197.218.177.69	TiD_list ∩ TiD_list = ICMP Echo Reply	1
195.115.218.108 ∪ 197.182.91.233= 195.115.218.108 ,197.182.91.233	TiD_list ∩ TiD_list = ICMP Destination Unreachable Port Unreachable	1

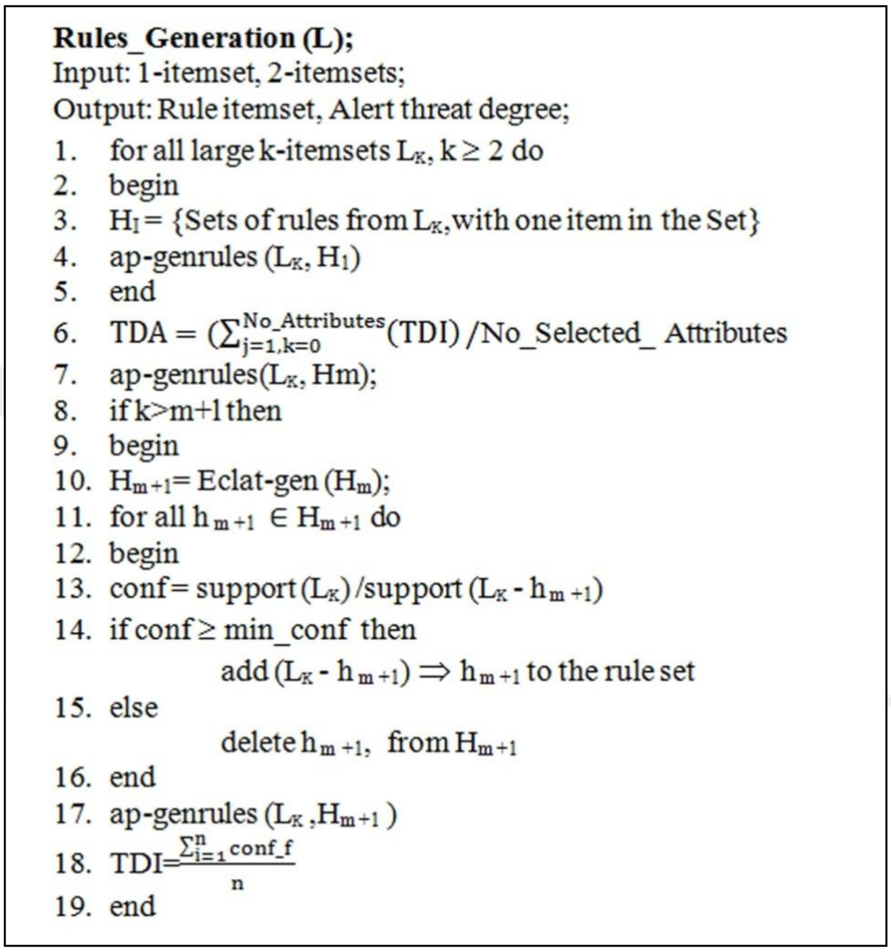
Figure 3.12 Results of Running Steps 8-15 of Eclat algorithm.

3.5.2 Generate Rules with Threat Degree sub-phase

Generate Rules with Threat Degree sub-phase is accomplished by using the improved Rules Generation algorithm to generate to generate rules for frequent itemsets, Threat degree for each item, and Threat degree for each alerts. The output of the first sub-phase represents the input for this sub-phase, while the output of the Generate Rules with Threat Degree sub-phase is the all alerts with its Threat Degree values; indeed, said alerts have been clustered by the last sub-phase of the ATE phase.

There are three components which make up this sub-phase, including: (1) Generating Rules for Frequent Itemset component, (2) Threat Degree of Items component, and (3) Threat Degree of Alerts component.

As such, these functions of the improved Rules Generation algorithm are passed through three processes. The improved algorithm is elaborated on in Figure 3.13, which explains said algorithm in detail.



```
Rules_Generation (L);
Input: 1-itemset, 2-itemsets;
Output: Rule itemset, Alert threat degree;
1.  for all large k-itemsets $L_K$, $k \geq 2$ do
2.  begin
3.  $H_1$ = {Sets of rules from $L_K$, with one item in the Set}
4.  ap-genrules ($L_K$, $H_1$)
5.  end
6.  TDA = $(\sum_{j=1,k=0}^{No\_Attributes}(TDI)$ /No_Selected_ Attributes
7.  ap-genrules($L_K$, Hm);
8.  if k>m+1 then
9.  begin
10. $H_{m+1}$= Eclat-gen ($H_m$);
11. for all $h_{m+1} \in H_{m+1}$ do
12. begin
13. conf= support ($L_K$) /support ($L_K - h_{m+1}$)
14. if conf $\geq$ min_conf then
            add ($L_K - h_{m+1}$) $\Rightarrow h_{m+1}$ to the rule set
15. else
            delete $h_{m+1}$, from $H_{m+1}$
16. end
17. ap-genrules ($L_K$ ,$H_{m+1}$ )
18. TDI=$\frac{\sum_{i=1}^{n} conf\_f}{n}$
19. end
```

Figure 3.13 Pesudocode of Rules Generation Algorithm

3.5.2.1 First Process: Rules of Frequent Itemsets Generation

There are two inputs for this process, namely 1-itemsets and 2-itemsets. These are generated by the Eclat algorithm, while the outputs of this process are the rules of itemsets, accompanied by the confidence values of these rules. This process starts from step 1 and ends with step 5, as shown in Figure 3.13. Firstly, these steps read the 1-itemsets and 2-itemsets, while the rules are generated only for the 2-itemsets, and the conditions are the same as those for the association rules. The gen-rules function is responsible for generating rules and calculating the confidence degree for each rule; the confidence degrees are extracted after the rules are generated based on equation (2.1). These rules are extremely significant, as they represent the degree of correlation between items in attributes. This equation will apply if the itemset is X, Y.

$$\text{Confidence } (\mathbf{X} \Rightarrow \mathbf{Y}) = \frac{\boldsymbol{Support}\ (\boldsymbol{X} \cup \boldsymbol{Y})}{\boldsymbol{support}(\boldsymbol{X})} \qquad (4.5)$$

For instance, the items 135.13.216.191 and 172.16.112.149 are 2-itemsets and have a support value which is equal to 1; moreover, their support values are 6 and 4 respectively. As such, in order to calculate the confidence degree for this rule (135.13.216.191 $\Rightarrow$ 172.16.112.149) equation (4.5) should be used, as follows:

$$\text{Conf } (135.13.216.191 \Rightarrow 172.16.112.149) = \frac{\boldsymbol{Support}\ (\mathbf{135.13.216.191, 172.16.112.149})}{\boldsymbol{Support}\ (\mathbf{135.13.216.191})}$$

$$= \frac{\mathbf{1}}{\mathbf{6}} = \mathbf{0.6666667}$$

This confidence degree is 0.666667, thus meaning that the rules in this set, along with their values, are stored in the same files of 1-itemset, which also contains the 2-itemsets.

3.5.2.2 Second Process: Calculation of the Threat Degree of Items (TDI)

This process is applied based on the equation in step 17, after extracting all rule sets and their confidence degrees for frequent 2-itemsets. In addition, this equation is proposed to calculate the TDI of each item. In essence, this equation is based entirely on the rules and confidence degrees that are extracted by the first process through the gen-rules function. This equation was designed based on the fact that erroneous faults are the most frequent alerts and all elements of a single alarm have a direct relationship with the other alert elements of the same attribute. More specifically, the value of confidence has a direct impact on the relationship between the items and their impact on each other. This value also depends on the support value and the impact of each item on the other; in actual fact, this represents the number of repeated items. Accordingly, the threat degree for each item is calculated based on equation (4.1), as follows:

$$TDI=\frac{\sum_{i=1}^{s} conf_I}{s} \tag{4.1}$$

Where:

$conf_I$: Denote the confidence degree of item.

S: Denotes the number of items present in the 2-itemsets.

To understand how the value of TDI for items is calculated, it is useful to explain the following example, as shown in Table 3.5.

Table 3.5 Calculating TDI of Items

S	Rule sets	confidence	conf summation	TDI
1	135.13.216.191-->172.16.112.149	0.6666667	0.6666667	0.6470588
2	135.13.216.191-->172.16.112.149	1	1.666667	0.6470588
3	135.13.216.191-->172.16.112.50	1	2.666667	0.6470588
4	135.13.216.191-->172.16.113.50	0.5	3.166667	0.6470588
5	135.13.216.191-->172.16.113.84	0.5	3.666667	0.6470588
6	135.13.216.191-->172.16.114.148	0.6666667	4.333333	0.6470588
7	135.13.216.191-->172.16.114.168	1	5.333333	0.6470588
8	135.13.216.191-->172.16.114.207	0.6666667	6	0.6470588
9	135.13.216.191-->172.16.114.50	1	7	0.6470588
10	135.13.216.191-->192.168.1.2	1	8	0.6470588
11	135.13.216.191-->194.27.251.21	0.3333333	8.333333	0.6470588
12	135.13.216.191-->194.7.248.153	0.1666667	8.5	0.6470588
13	135.13.216.191-->195.115.218.108	0.3333333	8.833333	0.6470588
14	135.13.216.191-->195.73.151.50	0.1666667	9	0.6470588
15	135.13.216.191-->196.227.33.189	0.3333333	9.333333	0.6470588
16	135.13.216.191-->196.37.75.158	1	10.33333	0.6470588
17	135.13.216.191-->197.182.91.233	0.6666667	11	0.6470588

The above table captures part of the result generated on the Monday of the second week from DARPA datasets; it shows that to calculate the proposed system, it is important to first calculate the threat degree of items 135.13.216.191 of attribute IPD.

$\sum_{i=1}^{s} conf_I$=(0.6666667+1+1+0.5+0.5+0.6666667+1+0.6666667+1+1+0.3333333

+0.1666667+0.3333333+0.1666667+0.3333333+1+0.6666667)=11

TDI of items 135.13.216.191= $\frac{11}{17}$ = 0.6470588

3.5.2.3 Third Process: calculation of the Threat Degree of Alerts (TDA)

The aim of this process is to extract the TDA of each alert after the TDI of each item has been extracted through the second process of the rule generation algorithm. The input for this process is the threat degree for each item, and thus each alert contains several items, and each item has a TDI value. The output of this process is the threat degree for each alert, while these values represent the input for the last phase of the proposed system (K-Means clustering). Table 3.6 below depicts the inputs of this process.

Table 3.6 TDI of part of alerts are processed

Alert_ID	G:S:RID	IPS	PS	IPD	PD
27	0.418229	172.16.113.84	P6264	192.168.1.20	P6667
36	0.320926	196.227.33.189	-p	172.16.112.100	-p

Each item has a TDI value, but to obtain the TDA for each alert, it is necessary to apply equation (4.2), as follows:

$$\mathrm{TDA} = \Big(\sum_{\mathrm{j=1,k=0}}^{\mathrm{No\_Attributes}} (\mathrm{TDI})\Big)/\mathrm{No\_Selected\_\ Attributes}$$

Where:

TDA: denotes threat degree of alerts

No_selected_ Attributes: denotes the number of features that are selected by user

This process is applied to extract the threat degree of alerts through two steps: the first step involves finding the TDI values summation for each alert. The second step involves determining how many attributes are selected by the user of the proposed system. An example is the best way in which to understand how this process is performed based on the data in Table 3.6. The results of this process have been obtained, as shown in Table 3.7.

Table 3.7 The TDI of alert 27

Alert_ID	G:S:RID	IPS	PS	IPD	PD
27	0.2367004	0.6666667	0.416667	0.2711112	0.5

As previously mentioned, alert 27 has several values that represent the threat degrees of items of attributes, and thus the final result for alert 27 is accomplished as follows:

$$\text{TDA}_{27} = \frac{0.2367004 + 0.6666667 + 0.416667 + 0.2711112 + 0.5}{5} = 0.3484232$$

3.5.3 Clustering TDA Values

The last phase of the proposed system is the K-Means algorithm. The inputs of this algorithm are the TDA values' attributes. Generally speaking, to perform the K-Means, it is important to provide two attributes' values, as this enables the algorithm to find the distance between each of the two attributes' values for the items in order to apply the Euclidian distance equation. However, the output of the second phase of the proposed system is only one attributes values, which in this case depends on the method from [71, 70]. Their method draws attention to the K-Means algorithm, and states that this will be transformed from one method to another. The first method involves finding similarities among the items by calculating the distances between those items either using Euclidean or Manhattan measures; in contrast, the new method involves calculating the mean value of k-distribution. Each attribute item is represented as a point in this distribution. This sub-phase produces two groups of distributions, and classifies the alerts based on the threat degree for these alerts. Figure 3.14 explains the K-Means clustering group for the TDA values.

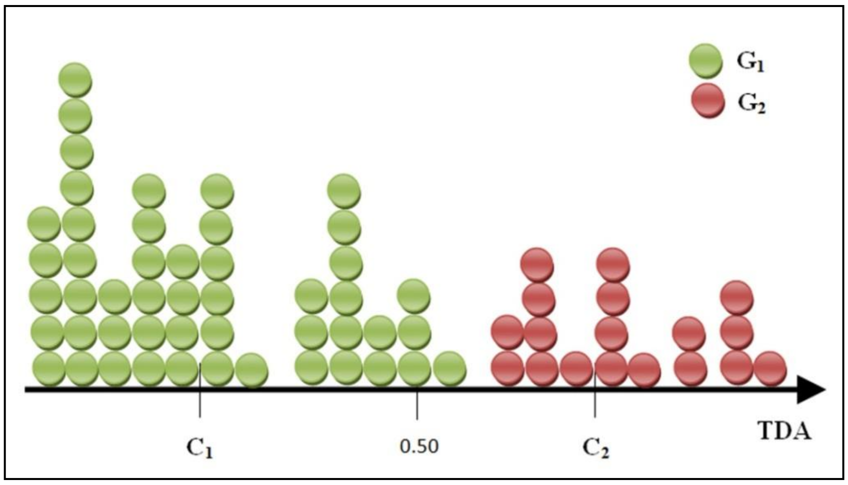


Figure 3.14 K-Means clustering group for TDA values

This algorithm clusters the alerts into two classes based on C_1 and C_2 values, which represent the threshold values of clustering; indeed, these values play a main role in determining the amount of reduction for positive alerts. The output of this algorithm consists of two groups of alerts, namely G_1 and G_2; these groups represent the false positive alerts and true positive alerts (real attacks), respectively. The following table (3.8) shows the parts of the results for this phase.

Table 3.8 Part of Final Results of last sub-phase of ATE phase

Alert_ID	G:S:RID	IPS	PS	IPD	PD	TDA	Clustering groups
1890	0.320926	197.218.177.69	-P	172.16.112.100	-P	0.09621212	G1
1894	0.308391	192.168.1.2	-P	192.168.1.1	-P	0.1050866	G1
1896	0.325058	192.168.1.1	-P	192.168.1.2	-P	0.513213	G2
1897	0.875787	129.53.216.18	P80	172.16.114.168	P8564	0.15	G1
1899	1:20258:1	172.16.112.149	P8813	134.205.131.13	P80	0.2547194	G1
1903	0.875787	137.245.85.134	P80	172.16.112.149	P9786	0.4485236	G2
1910	0.491782	172.16.112.100	P20	135.13.216.191	P4678	0.10307	G1

CHAPTER 4

RESULT AND DISCUSSION

4.1 Introduction

This chapter will discuss both the evaluation of the phases and all of the results obtained for each phase of the proposed system. The chapter is arranged as follows: Section 4.2 presents a detailed explanation of the Experiment Environment. Section 4.3 summarizes the Evaluation Criteria of the Proposed System. Following this, Section 4.4 presents the experiments of Evaluation of FDA Phase**.** Finally, Section 4.5 illustrates the experiments evaluation of the ATE phase.

4.2 Experiment Environment

All the results presented here have been obtained by the proposed system, which includes three phases, namely the filtering duplicate alerts phase, the alert threat evaluation phase, and the clustering phase. Prior to discussing the results of the proposed system, it is best to explain the hardware and software specifications, as well as the datasets that have been used to implement the proposed system.

4.2.1 Hardware and Software Specifications

There are certain specifications which are used to test the performance of the proposed system. These specifications include software and hardware respectively, as seen below:

- Operating system: Windows 7 Ultimate.
- Compiler: Visual Studio 2015 Professional.
- Database file: Microsoft Office Excel 2007.
- CPU: Intel ® Core TM i5-4210U CPU@1.70 GHZ.
- Memory: 4.00 GB.

4.2.2 Datasets

The DARPA 99 dataset has been used in this thesis to assess the performance of the proposed system, and specifically to assess the cogency of said system. The total number of alerts used from this dataset stands at 59,720, spread over five weeks. This data has been collected by using Snort, and the output alerts are saved in a Snort log file. This dataset has been explained in detail in Chapter 2. The present thesis concentrates solely on the full mode alerts, because these alerts have all of the significant attributes. This dataset has been used both partially and entirely so as to compare it with other studies which have used part of or all of the dataset.

4.3 Evaluation Criteria of the Proposed System

There are several metrics, including number of attributes and reduction of false positive rate, which are used to evaluate the phases of the proposed system.

4.3.1 Number of Attributes

Before the proposed system is used, the dataset is checked in order to choose the highest weight attributes; this step is used only once. In actual fact, this step provides the flexibility for systems analysts to select the number of attributes. The proposed system also provides the systems analysts with the ability to increase the number of attributes which are used. Table 4.1 presents a comparison of relevant studies based on the number of attributes used.

Table 4.1 The comparison based on No. of Attributes used

Methods	No. of Attributes used in reduction systems	Elasticity of increasing No. of Attributes	Reduction Ratio
Hachmi and Limam (2013)	5 Attributes	NO	92.8%
Al-mamory and Zhang (2010)	5 Attributes	NO	82%
Al-shammari et al. (2007)	6 Attributes	NO	90.92%
The proposed system 2017	More than seven attributes based on attributes selected by analyst	Yes	97.98%

The above table shows the different numbers of attributes used in their systems. These systems do not allow the user to increase the number of attributes or change the attributes which have been used. Although the attributes are available, the user does not have the ability to handle any effective attribute from this selection. Indeed, this situation results from a lack of elasticity on the part of these systems to deal with all types of attributes. Finally, this makes it difficult to deal with such systems.

Of particular note here are the studies by Hicami and Limam (2013) and Al-shammari et al. (2007); although these researchers chose five and six attributes – not lower than the number of attributes selected by other studies concerning antecedents' attributes – their methodology led to a good reduction in the number of false alerts. This is due to their choice of better attributes, which have a substantial impact, as well as their use of an efficient method, which resulted in a highly-effective reduction process. As for Al-mamory and Zhang (2010), these authors chose the same number of attributes, but generated a reduction ratio less than those produced by the above-mentioned studies. The reason for this is either the fact that they used inappropriate methods, or that they employed attributes which have less impact.

4.3.2 Reduction Ratio

This is measured by counting the ratio of the number of alerts that represent the false alerts which are labeled G1 in the output file to the total number of alerts in the input file which denotes as M. The ratio of the reduction alerts is measured based on equation (4.1).

$$\text{Rr} = \frac{No.of\ FP}{M} * 100 \qquad (5.1)$$

The percentages of false positive alerts vary significantly based on the clustering thresholds that are selected by the systems analyst or systems administrator to fit with the best results.

4.4 Evaluation of FDA Phase

This phase is implemented implicitly as an integral part of the proposed system; indeed, this stage cannot be implemented separately from the other phases of the proposed system – an issue which has been explained further in Chapter 3. This phase involves removing the duplicated alerts that result from benign traffic activities and which have been incorrectly classified by intrusion-detection systems. This problem has been addressed in detail in Chapter 2. To evaluate this phase, three weeks of the DARPA dataset are used to apply the methods used in this phase to filter said alerts.

The results of the three weeks of the DARPA 99 dataset are shown in Figure 4.1, which presents the percentages of the alerts filtered by this phase for each day of these three weeks.

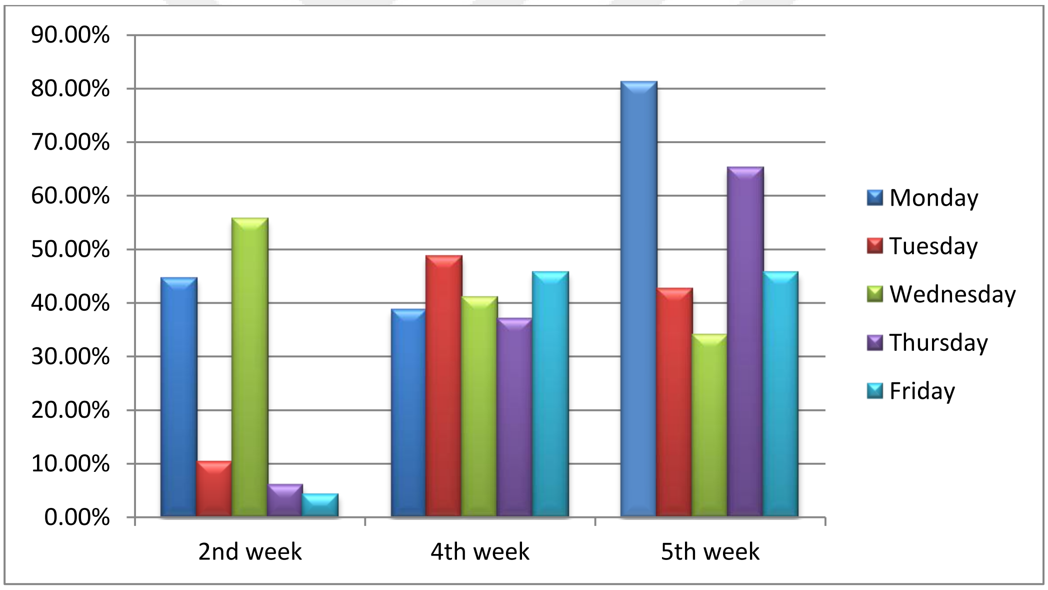


Figure 4.1 Results of the first phase (FDA phase)

Table 4.2, below, explains the total number of alerts for each week which were obtained by this phase after and before implementation. In addition, the table shows the percentages of alerts filtered for each week.

Table 4.2 Detailed results of FDA phase

Weeks	Input Alerts	Output Alerts	No. of alerts filtered by this phase	Filter duplicated Alerts Ratio %
Second week	24005	20294	3711	15.46%
Forth week	8053	4609	3444	42.77%
Fifth week	14065	5108	8957	63.68%
Total	46123	30011	16112	34.93%

This phase generates a reduction ratio of 34.93%, which is very good, as it proves that this phase is no less important than the second phase of the proposed system in reducing false alarms by removing the duplicated alerts. Thus, the results of the Filter duplicated alerts phase of the proposed system are explained in order to illustrate the magnitude of the effect of this phase on the overall performance of the proposed system; these results are also explained to prove the effectiveness of the method used in this phase and to indicate the contribution of this phase in reducing false alarms.

4.5 Evaluation of ATE phase

The phases of the proposed system are based on data which is generated by past phase. Thus, the ATE phase is based on the output data from the Filter duplicated alerts phase. As explained in Chapter 3, this phase calculates the threat degree by applying two sub-phases, namely the threat degree for items sub-phase, and the threat degree for alerts sub-phase. This phase is proposed to evaluate each alert by calculating these two values (TDI and TDA). This aforementioned phase is tested by utilizing five weeks of the DARPA 99 dataset for five weeks (first week, second week, third week, fourth week, and fifth week).

The parameters that are used in these experiments are as follows: firstly, the data, which is from five weeks of the DARPA 99 dataset, consists of 59,720 alerts.

Secondly, the No. of Attributes selected stands at six (the systems analyst can change the No. of Attributes). Thirdly, for the min_sup threshold, the minimum support used is equal to 1, as explained in detail in Chapter 3 (the systems analyst can change this threshold). Finally, with regard to the clustering thresholds, in this experiment several thresholds are used, including: (0.10-0.40), (0.20-0.50) and (0.25-0.75); moreover, the systems analyst can change these threshold values. Equation (4.2) is applied to calculate the average alert reduction ratio for the weeks of the dataset for each experiment; this makes it possible to obtain the best threshold clustering, which presents the highest and best reduction ratio.

$$\text{Average alert reduction ratio} = \frac{W_1 + W_2 + \ldots\ldots + W_N}{N} \tag{4.2}$$

Experiment 1: As explained in past chapters, it is the clustering threshold that plays a significant role in the classification of alerts based on the threat value of each alert. Several results are obtained by using several different thresholds and testing the data (second week, fourth week, and fifth week) from the DARPA 99 dataset. These weeks contain alerts concerning real attacks. Figure 4.2 depicts the results of the second week, third week and fifth week by using a threshold value equal to (0.10-0.40).

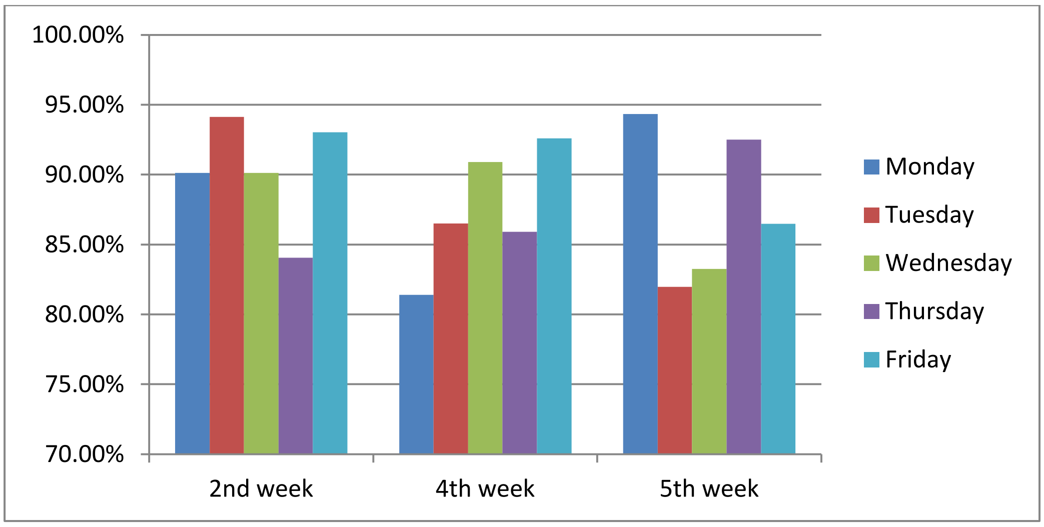


Figure 4.2 Results of (0.10-0.40) Threshold Value

The above figure shows the results of the proposed system, with a threshold value equal to (0.10-0.40). The highest percentage of reduction of false alerts by the proposed system is equal to 94.34%, while the lowest percentage of reduction is equal to 81%. By applying equation (4.2), an average alert reduction ratio of 88.48 is obtained. These results are more apparent in each of the following tables (Table 4.3 illustrates the details of total alerts in the second week dataset, while Table 4.4 explains the details of alerts in the fourth week dataset, and Table 4.5 presents details of the results in the fifth week dataset).

Table 4.3 The detailed results for the second week with (0.10-0.40) threshold value

Second week	No. Alert	No. Attack	% of attack	No. False positive	% of False positive
Friday	11122	775	6.968171	10347	93.031829
Monday	1550	153	9.870968	1397	90.129029
Tuesday	4121	242	5.872361	3879	94.127637
Wednesday	3401	336	9.879448	3065	90.120553
Thursday	3811	608	15.95382	3203	84.046185
The average of FP for second week			90.2910466		

Table 4.4 The detailed results for the fourth week with (0.10-0.40) threshold value

Fourth week	No. Alert	No. Attack	% of attack	No. False positive	% of False positive
Friday	2598	192	7.3903	2406	92.609697
Monday	600	112	18.66667	488	81.333333
Tuesday	1314	178	13.54642	1136	86.453574
Wednesday	1813	165	9.100938	1648	90.899062
Thursday	1728	244	14.12037	1484	85.879629
The average of FP for fourth week			87.435059		

Table 4.5 The detailed results for the fifth week with (0.10-0.40) threshold value

Fifth week	No. Alert	No. Attack	% of attack	No. False positive	% of False positive
Friday	1604	214	13.52868	1387	86.471319
Monday	5511	312	5.661405	5199	94.338595
Tuesday	1408	254	18.03977	1154	81.960225
Wednesday	1457	244	16.74674	1213	83.253258
Thursday	4085	306	7.49082	3779	92.509180
The average of FP for fifth week			87.7065154		

Experiment 2: Figure 4.3 presents the results for the same weeks of datasets which have been used in previous experiments, but with the (0.20-0.50) threshold value; this is done in order to test the performance of the proposed system with more than one threshold value. In addition, this is carried out in order to obtain the best threshold value when it comes to reducing the number of false alerts by the largest amount.

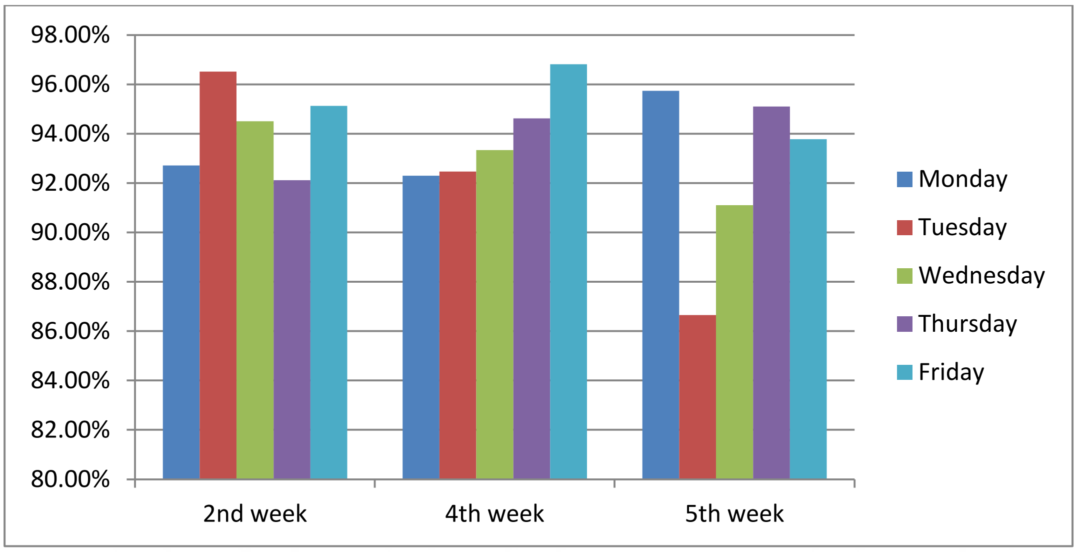


Figure 4.3 Results of (0.20-0.50) threshold value

Figure 4.3 explains the results of the proposed system, with a threshold value equal to (0.20-0.50). The highest percentage of reduction of false alerts by the proposed system is equal to 96.81%, while the lowest percentage of reduction is equal to 86.65% in this experiment. By applying equation (4.2), an average alert reduction ratio of 93.6% is obtained. These results are more pronounced in each of the following tables (Table 4.6 illustrates the details of total alerts in the second week dataset, Table 4.7 explains the details of alerts in the fourth week dataset, and Table 4.8 presents the details of results for the fifth week of the dataset).

Table 4.6 The detailed results for the second week with (0.20-0.50) threshold value

Second week	No. Alert	No. Attack	% of attack	No. False positive	% of False positive
Friday	11122	542	4.873224	10580	95.126777
Monday	1550	113	7.290322	1437	92.709678
Tuesday	4121	144	3.494297	3977	96.505701
Wednesday	3401	187	5.498383	3214	94.501614
Thursday	3811	300	7.87195	3511	92.128050
The average of FP for second week			94.194364		

Table 4.7 The detailed results for the third week with (0.20-0.50) threshold value

Fourth week	No. Alert	No. Attack	% of attack	No. False positive	% of False positive
Friday	2598	83	3.194765	2515	96.805232
Monday	600	46	7.666667	554	92.333334
Tuesday	1314	99	7.534247	1215	92.465752
Wednesday	1813	121	6.674021	1692	93.325978
Thursday	1728	93	5.381944	1635	94.618058
The average of FP for fourth week			93.9096708		

Table 4.8 The detailed results for the fifth week with (0.20-0.50) threshold value

Fifth week	No. Alert	No. Attack	% of attack	No. False positive	% of False positive
Friday	1604	100	6.234414	1504	93.765586
Monday	5511	235	4.264199	5276	95.735800
Tuesday	1408	188	13.35227	1220	86.647725
Wednesday	1457	130	8.922443	1327	91.077554
Thursday	4085	201	4.920441	3884	95.079559
The average of FP for fifth week			92.4612448		

Experiment 3: The last threshold value in these experiments is (0.25-0.75), which generates the highest reduction ratio. This experiment with this threshold value reduces the number of alerts, and by applying equation (4.2), an average alert reduction ratio of 97.2% is obtained for three weeks of the DARPA 99 dataset. Figure 4.4 illustrates the reduction ratio for these weeks. Experiment 3 is conducted with all five weeks of the DARPA 99 dataset.

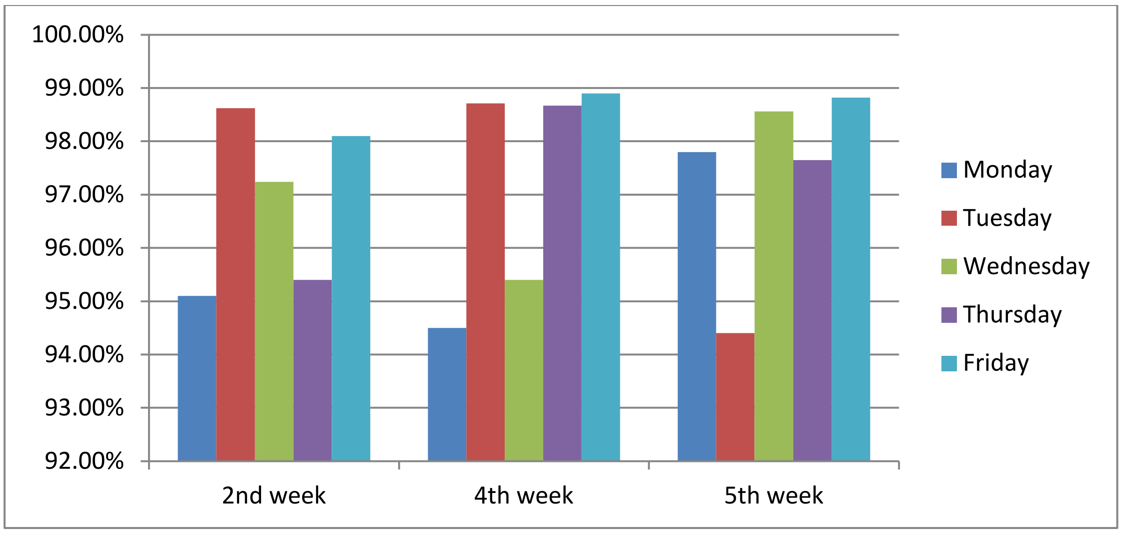


Figure 4.4 Results of (0.25-0.75) Threshold Value

The clustering threshold (0.25-0.75) is the best threshold value, and provides the highest reduction ratio when compared with other threshold values used to test the performance of the proposed system, through using the first week and third week from DARPA 99 datasets which are free from attacks; this means that there are no attacks in either of these two datasets (100% false positive alerts). Table 5.9 illustrates the obtained results by using the phases of the proposed system.

Table 4.9 Reduction Ratio Results after using (0.25-0.75) Threshold value for five weeks of DARPA Dataset.

Week	No. of Input Alerts	No. of Output Alerts	Reduction ratio
First week	7,293	35	99.486472%
Second week	24,005	617	96.8814784%
Third week	6,304	36	99.403095%
Fourth week	8,053	185	97.2328514%
Fifth week	14,065	336	97.443535%
Total	**59720**	**1209**	**97.98%**

In particular, these two weeks generate an extremely high rate in reduction ratio, as shown in Table 4.9. The number of alerts used in the first, third, and fourth weeks do not exceed the number of alerts used in other weeks; despite this, the highest reduction percentage has been obtained. Figure 4.5 explains the reduction ratio for these five weeks, with threshold (0.25-0.75), as follows.

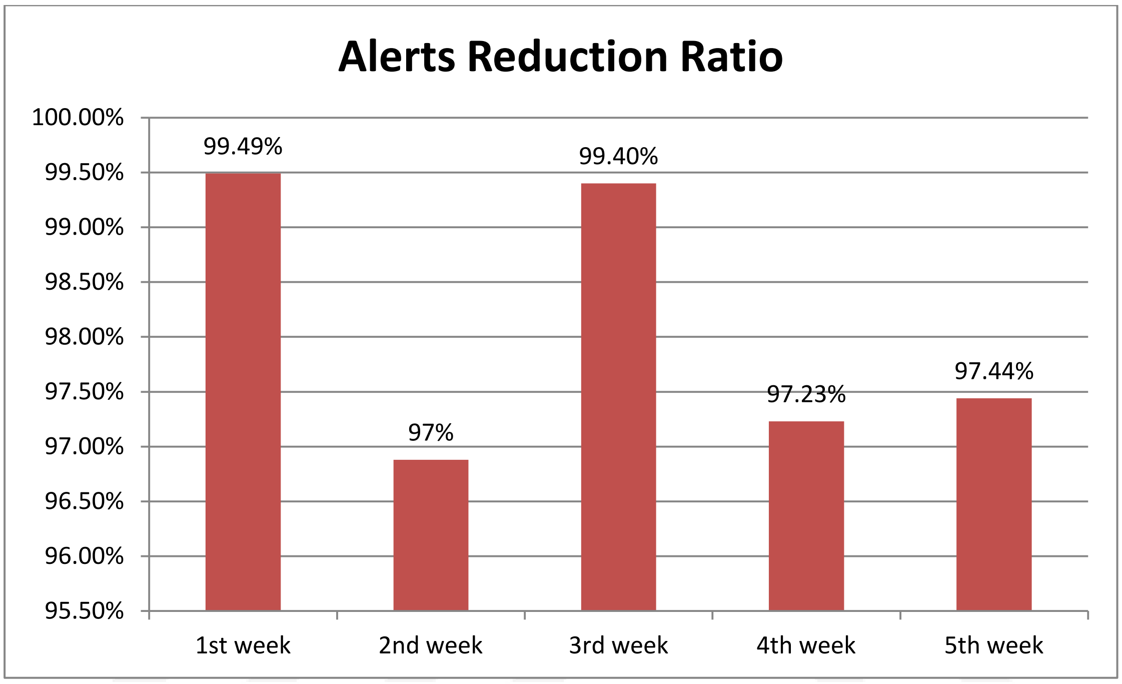


Figure 4.5 Reduction Ratio Results after using (0.25-0.75) Threshold value for five weeks of DARPA Dataset.

In particular, the main goal for this phase is to reduce the FP alerts. This goal is achieved when these algorithms of the ATE phase are applied. With regard to this method, and as mentioned and explained in detail in Chapter 3, this phase depend on several parameters. The clustering threshold in the second phase plays a very effective role in reducing the redundant alerts. Moreover, the time-based similarity between alerts' items based on time stamp is also an effective factor in reducing FP alerts. Many of the duplicated alerts are disposed of in the first phase of the proposed system. These combined factors lead to a reduction rate increase, which reaches 99.48% for the first and third weeks.

Many researchers have proposed a number of approaches to reduce alerts, as mentioned previously in Chapter 2, section 2.2.3. Table 4.10 shows a comparison between the approach put forth by Hachmi and Limam (2013), and the approach proposed in the present thesis; for this comparison, the first day from each the fourth week and the fifth week of the DARPA 99 dataset is used.

Table 4.10 Comparison based on Reduction Ratio between the thesis approach with another approach by using the first day from each the fourth week and the fifth week from dataset

Weeks	The No. of input alerts	The No. of output alerts	Reduction Ratio
Hachmi and Limam (2013)	956	49	94.83%
Proposed Approach 2017	6111	140	97.71%

Table 4.10 shows the differences between the reduction ratio of this thesis' approach and the approach put forth by Hachmi and Limam (2013); the table shows that the reduction ratio produced by the latter authors is less than the reduction ratio of the proposed system, because their approach suffered from a lack of accuracy for several reasons. The main reason is that they did not identify the all types of attacks alerts, although their approach is intended to concern only to identify four types of attacks alerts. When comparing the number of alerts generated by the proposed system and the alerts generated by the Hachmi and Limam method, it appears that this method still suffers with issues, and that the FP problem of IDSs has not been solved efficiently because they did not use the all available dataset that contained on the various types of attacks alerts. This is clear when looking at the input alerts and alerts resulting from this approach, which still exceed the systems analyst's ability to verify them; in contrast, the proposed approach of this thesis is based on the effective and efficient parameters, including the similarity between items, as well as the timestamp

to eliminate the duplicate alerts. In addition to this, the TDI, TDA, and the clustering threshold are used to eliminate the redundant alerts. All of these parameters lead to a high reduction ratio.

To compare the results of the proposed approach with other previous approaches, Table 4.11 presents several comparisons. These researchers used all five weeks of the DARPA 99 database, while Al-shammari et al. (2007) used only two weeks from the DARPA dataset.

Table 4.11 Comparisons between the proposed system and several previous approaches

Weeks	The No. of input alerts	The No. of output alerts	The No. of weeks	Reduction Ratio %
(Al-shammari et al., 2007)	27,877	7,118	4th & 5th weeks	90.2%
(Jie Ma et al., 2008)	__	__	Five weeks	90%
(Al-Mamory et al., 2010)	233,615	42,051	Five weeks	82%
Proposed system 2017	59,720	1,209	Five weeks	97.98%

Table 4.11 explains that Al-shammari et al. (2007) produced a good reduction rate, despite the number of alerts used when compared to the number of alerts used by other researchers. However, their reduction rate was very close to the reduction rate obtained by Jie Ma et al. (2008). In contrast, Al-Mamory et al. (2010) obtained a questionable ratio of reduction, despite the high number of input alerts used. Figure 4.6 illustrates these comparisons more clearly.

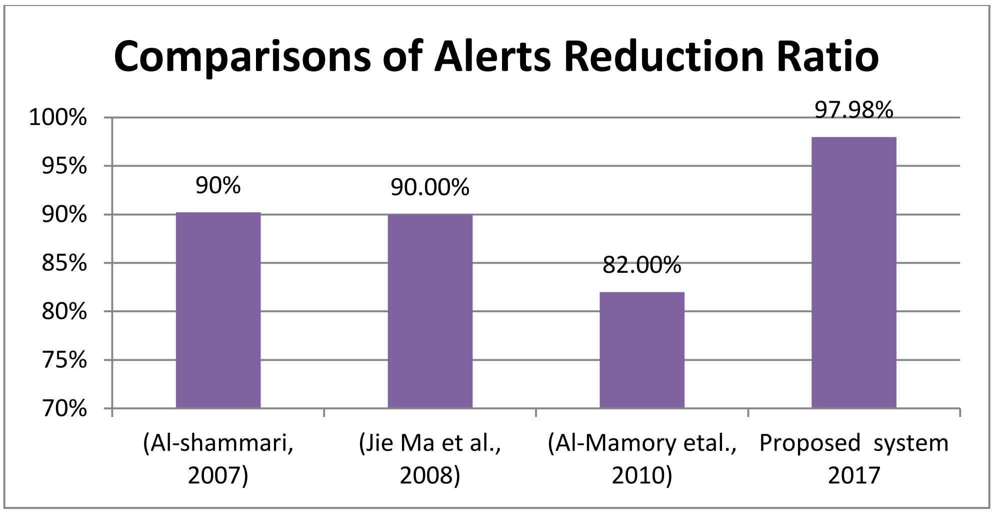


Figure 4.6 Comparisons of Reduction Ratio of the Proposed System with other Approaches

These results from the other researchers indicate that their approaches did not perform well in terms of obtaining high ratios of the reduction process. In addition, these approaches did not depend on suitable parameters, including selecting the right attributes which have a more significant effect on the reduction process. Instead, their approaches relied on selecting the attributes randomly, without depending on the weight of the attributes, as is the case with our proposed system. Finally, they did not seek to calculate the threat degree of each alert in order to obtain the highest rate of reduction of false alarms, as is also the case with our proposed system. For these reasons, our proposed system obtained the highest reduction rate; because it took into account all of these factors in various two phases of the proposed system.

4.6 Complexity of the proposed system

In order to explain the efficiency of the proposed system, its complexity will be discussed in detail, including the time needed to implement this system and the demands which this system places on the CPU. In particular, the complexities are explained to demonstrate the correlation between the proportion of data volume increments and the increments in the exhaustion of resources of the environment.

In actual fact, the execution time describes how much time the system needs to perform its functions. The performance of the proposed system was tested by using datasets of different sizes, since each day of the DARPA 99 database contains various sizes of datasets. For instance, when operating the proposed system, in order to execute its process (eliminate the duplicated alerts and evaluate the redundant alerts to diminish FP alerts) on different sizes of data, the largest and lowest volumes of data are used to show the complexity of the system. Table 4.12 displays the details of the experiment implementation for the two days selected (the fifth day from the second week and the second day from the fifth week) from the DARPA dataset.

Table 4.12 the detailed results of experiment implementation for two days selected from DARPA dataset to show the complexity of the proposed system

Data	No. of input alerts	Parameters	CPU	Memory	Time
Fifth day of second week	11,122	Minsup =1, 7 attributes, and (0.25-0.75) clustering threshold	(7%-10%)	179,458K	2.09M
Second day of fifth week	1,408		(2%-4%)	132,784K	1.12M

The information in the table above shows that the proposed system has been implemented with high efficiency, due to its minimal consumption of the hardware's total CPU capacity. In addition, this experiment demonstrates the efficiency of the system's performance, as it shows that all system functions are implemented in a speedy manner. Finally, this experiment did not consume much of the storage space; indeed, very little storage space was used by the proposed system.

It is important to mention **why this proposed system does not reduce the false alarms by 100%.** All the methods suggested by previous researchers included error ratios; with regard to the system proposed in this thesis, although it produces a high percentage in terms of reducing the number of false alarms, as well as high efficiency, it also generates an error rate. The DARPA database contains two types of percentages when it comes to reduction ratio, both of which are adopted by the DARPA website and Tjhai; as such, the error rate of the proposed system will be calculated.

It is clear that there exist two types of percentages when it comes to the reduction ratio, and these percentages are adopted by the DARPA website. The first and third weeks are both labeled ‘attack free’, meaning that all alerts are false alerts, equating to a percentage of 100% (DARPA, 2011); in contrast, the second type of percentage yields a figure of 99%, which is referred to by the researcher (Tjhai, 2008). The DARPA website provides information related to the DARPA 99 dataset, and classifies this dataset into two groups based on the attack labels: the first group is the ‘attack free’ group, which consists of the first and third weeks, while the second group is the ‘attack label’ group, which consists of the remaining weeks in the dataset.

The reduction ratio of the proposed system for the first group is 99.48%. In fact, this ratio contains the average of the error rate, which does not exceed a ratio of 0.52%. This result provides evidence, and is a strong indication that the methodology of the proposed system is credible; it also reveals the truthfulness in reducing the false positive alerts.

The second group is the ‘attack label’ group, which means that this dataset group contains real attacks. The results of the three weeks are obtained, and yield a rate ratio reduction of 97.53%. In actual fact, this obtained reduction ratio is very good, because the total ratio of the false positive alerts is 99% (Tjhai, 2008). Thus, the error rate of the proposed system for this group does not exceed 1.47%.

The error rate in this system results from some erroneous alerts that have not been repeated many times, but which come in the form of one or two alerts, and so they were evaluated as alerts generated from a real attack; indeed, this occurred in the first week and the third week, while these weeks contain data which is free from real alerts. Depending on the information provided by the DARPA website (DARPA, 2013), there are certain alerts which result from a real attack, but alerts have been repeated for this attack, and sometimes these alerts are repeated for more than 70 alerts from a single real attack. These alerts are classified as false alerts by the proposed system. The principle which has been adopted in the design of this proposed system is that the wrong alarm is the most frequent alert and the degree of alert threat is evaluated based on this theory. These alerts are real attack alerts, as is the case in the second, fourth and fifth weeks; indeed, this is evidenced by the results obtained. Table 4.13 shows the analysis of the real attack alerts for the fourth week and displays the number of repeated real attacks alerts (DARPA, 2013), all of which have been evaluated by the proposed system as false alerts; indeed, this has affected the rate of minimizing erroneous alerts.

Table 4.13 Real attack alerts for the fourth week of the DARPA 99 dataset

Days of fourth week	Alert ID	No. of alerts	Attacks name	Category
W4 D1 3-29-1999	41.162715	128	Portsweep	PROBE
W4 D2 3-30-1999	42.155148	73	mailbomb	DOS
W4 D3 3-31-1999	43.191217	206	Snmpget	R2L
W4 D4 4-01-1999	44.080757	88	ipsweep	PROBE
W4 D5 4-02-1999	45.192523	271	ipsweep	PROBE

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

5.1 Introduction

In the last three chapters, focus has been on the filtering duplicated alerts (FDA) phase and the alert threat evaluation (ATE) phase; their results and details regarding their implementation have been rigorously illustrated. This chapter includes Section 5.2, which draws a conclusion about the system as a whole, while discussion will also focus on the goals of the thesis that have been accomplished. Section 5.3 discusses suggestions and future work.

5.2 Conclusion

The system in this thesis is proposed and has been designed to reduce the number of false positive alerts by filtering duplicate alerts and evaluating redundant alerts with a view to improving and obtaining high quality networks security. Specifically, this proposed system has addressed all objectives of the present thesis, and has produced the following explanation:

The first objective of this thesis is (To propose a new filtering algorithm which removes the duplicate alerts and which depends on specific attributes as well as the time stamp attributes and other attributes). This objective is achieved through the Filtering Duplicate Alerts phase in the proposed system. This phase is based on two proposed sub-phases, which aim to reduce the number of false positive alerts. This FDA phase generates a reduction ratio of 34.93%, which is very good because it proves that this phase is no less important than the second phase of the proposed system in reducing false alarms by removing the duplicated alerts.

To reduce the number of false positive alerts that remain from the first phase, the Alert Threat Evaluation phase (ATE) has been proposed. Thus, the second phase has been successful in addressing the second thesis objective, namely to improve existing algorithms so that they are more suitable when used with multiple items of a single attribute. These improved algorithms will be capable of calculating the threat degree of each item (TDI) and the threat degree of each alert (TDA) so as to evaluate the TDA values of alerts and to cluster them. All of these steps in this phase aim to remove the redundant alerts.

The first sub-phase of the second phase of the proposed system is accomplished by performing the Eclat algorithm. The aim of said algorithm is to generate itemsets, which is considered very important in evaluating the item threat degree in the upcoming sub-phase. Indeed, this sub-phase has been applied to the data to find the strength of the relationship between each item and the other items in the single attribute. In actual fact, this sub-phase has greatly contributed to enhancing the efficiency of the proposed system by significantly contributing to the evaluation of the threat level to obtain the best results.

The second sub-phase of this phase is concerned with generating rules for frequent itemsets and extracting the threat degree of each alert. The generated rule algorithm has been improved so that it can calculate the threat degree of the items (TDI) and the threat degree of alerts (TDA) implicitly, and by using the alternative way of K-Means algorithm to clustering the TDA values; thus this clustering method in the second phase played a very effective role in reducing the redundant alerts. All these processes have contributed significantly and effectively to reducing false alerts by a very large percentage

Finally, by using the first week and third week of the DARPA 99 dataset, it has been possible to test the efficiency of the system. The system has generated a reduction ratio of 99.48%, which is the best possible result when considering the fact that these two weeks consist solely of false alerts (free attack label). Consequently, this proposed system has generated a high reduction ratio of 97.98%, as all weeks of the DARPA dataset are used. Based on the results which have been obtained, the performance of the ATE phase has been proven.

5.3 Future Work

The proposed system is evaluated and implemented by using the DARPA 99 dataset, which is generated by using Snort; this system is proposed to work with any data that is generated by other types of IDSs, such as BRO, etc. Despite the better results achieved by this proposed system, it is better to design a new IDS system which takes into account the method proposed in this thesis, so as to achieve further efficiency.

This proposed method can be improved by altering it so that it can be implemented parallelly. This capability can be developed by distributing the execution of the second phase of the proposed system (mining frequent itemsets and generating rules for frequent itemsets) to the parallel processing threads. This development can lead to increased time efficiency and performance speed.

In addition to the method proposed in this thesis, it is also possible to use machine learning algorithms to improve the performance of the proposed system and to increase the efficiency of the proposed system, so as the problems of IDSs can be addressed, especially to get rid of the problem of repeated alerts of the real attacks that resulted in the error rate in the implementation of the proposed methodology.

REFRENCES

[1] McAfee Labs (2013). McAfee Labs Threats Report. available in https://www.mcafee.com/us/resources/reports

[2] Julisch, K. Dealing with false positives in intrusion detection. available in " http://www.raid-symposium.org/", (2000).

[3] Axelsson, S. 1999. The base-rate fallacy and its implications for the difficulty of intrusion detection. In *CCS ’99: Proceedings of the 6th ACM conference on Computer and communications security*, 1-7, New York, NY, USA. ACM.

[4] Manganaris, S., Christensen, M., Zerkle, D., and Hermiz, K. 2000. A data mining analysis of rtid alarms. *Comput. Netw.*, 34(4), 571-577.

[5] Julisch, K. (2003). Using root cause analysis to handle intrusion detection alarms, PhD dissertation, University of Dortmund.

[6] Elshoush and Osman. (2011). Alert correlation in collaborative intelligent intrusion detection system-A survey. *Applied Soft Computing Journal*, 11, 4349-4365.

[7] Tjhai G. C. (2011). Anomaly-Based Correlation of IDS Alarms, PhD thesis, The University of Plymouth, UK.

[8] Magi, F., Matteucci, M., and Zanero, S. (2009). Reducing false positives in anomaly detectors through fuzzy alert aggregation, Information Fusion. 10, 300-311.

[9] Adnan, A. H. (2009). Multithreaded scalable matching algorithm for intrusion detection system. University Sains Malaysia, PhD Thesis.

[10] El-Taj, H., Abouabdalla, O., Manasrah, A., Al-Madi, A., Sarwar, M.I., and Ramadass, S. (2010). Forthcoming aggregating intrusion detection system alerts

framework. In Emerging Security Information Systems and Technologies (SECURWARE), 2010 Fourth International Conference. 40-44. IEEE.

[11] Axelsson, S. (2000). The Base-Rate Fallacy and the Difficulty of Intrusion Detection. 3(3):186–205.

[12] Tjhai, G. C., Papadaki, M, Furnell S. M., and Clark N. L. 2008. The problem of false alarms: Evaluation with Snort and DARPA 1999 Dataset. Submitted to TrustBus 2008, Turin, Italy, 1-5 September 2008.

[13] Mannila, H., Toivonen, H., and Verkamo, A. I. 1997. Discovery of frequent episodesin event sequences. *Data Mining and Knowledge Discovery,* 1(3), 259-289.

[14] Bishop, M. A. 2002. *The Art and Science of Computer Security*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[15] McHugh, J. 2001. *Intrusion and intrusion detection*. Springer Berlin / Heidelberg

[16] S. Chebrolu, A. Abraham, and J. P. Thomas, _Feature deduction and en-semble design of intrusion detection systems,_ Computers and Security, 2004.

[17] Hackmageddon.com, _Cyber attacks statistics,_ http://hackmageddon.com/2013-cyber-attacks-statistics/, Aug 2013.

[18] Smaha, S. 1988. Haystack: an intrusion detection system. *Aerospace Computer Security Applications Conference, 1988., Fourth,* 37-44.

[19] Zhou Wei, L. and Zheng Shun-Zheng, Y. 2006. An HTTP flooding detection method based on browser behavior. Proc. Of 2006 International Conference on Computational Intelligence and Security. Guangzhou, 2, 1151-1154.

[20] Szmit, Maciej, and Szmit A. (2012). Usage of modified holt-winters method in the anomaly detection of network traffic: case studies. *Journal of Computer Networks and Communications*, 2012.

[21] Deng, J. D, and Purvis, M. K. 2011. Multi-core application performance optimization using a constrained tandem queuing model. *Journal of Network and computer Applications*, 34, 1990-1996 doi:10.1016/j.jnca. 2011.07.004.
[22] Bace, R. G. 2000. *Intrusion detection*. Macmillan Publishing Co., Inc., Indianapolis, IN, USA.

[23] Xu, D. and Ning, P. (2008). Correlation analysis of intrusion alerts. *Intrusion Detection Systems, Advances in Information Security*, 38, 65-92.

[24] Hoang, X. D., Hu, J., and Bertok, P. 2009. A program-based anomaly intrusion detection scheme using multiple detection engines and fuzzy inference. *Journal of Network and Computer Applications*, 32, 1219-1228.

[25] Morin, B., Me, L., Debar, H., and Ducasse, M. 2002. M2D2: a formal data model for IDS alert correlation. 5th international symposium: recent advances in intrusion detection, 115-137 Springer-Verlag Berlin, Heidelberg.

[26] Paxson, V. 1999. Bro: a system for detecting network intruders in real-time. *Comput. Netw.*, 31(23-24), 2435-2463.

[27] Kumar, S. *Classification and detection of computer intrusions*. PhD thesis, West Lafayette, IN, USA, 1995.

[28] Ilgun, K. 24-26 May 1993. Ustat: a real-time intrusion detection system for unix. *Research in Security and Privacy, 1993. Proceedings., 1993 IEEE Computer Society Symposium on*, 16-28.

[29] Tanenbaum, A. S. 1996. *Computer networks*. Englewood Cliffs: Prentice-Hall, |c1996, 3rd ed., international edition.

[30] Rafeeq Ur Rehman "Intrusion detection system with snort – advanced IDS techniques using Snort, Apache, MySQL, PHP, and ACID" 1st printing, PTR, USA, ISBN: 0-13-140733-3. 66-70, 2003.

[31] Snort. http://snort.org/ (Last visited March 2008).

[32] Syngress, How to cheat at conjuring open source security tools. Syngress, 2007.

[33] Paxson, V. 1999. Bro: a system for detecting network intruders in real-time. *Comput.Netw.*, 31(23-24), 2435-2463.

[34] Berry, M. and Linoff, G. 1999. Mastering Data Mining: the Art and science of Customer Relationship Management. John Wiley & Sons, Inc.

[35] Chen, T. S. J. and Kao, Y. H. (2010). A Novel Hybrid Protection Technique of Privacy-Preserving Data Mining and Anti-Data Mining. *Information Technology Journal*, 9:500-505. Doi:10.3923/itj.2010.500.505.

[36] Cios J. K., Pedrycz W., Swiniarski R. W., and Lukasz Andrzej Kurgan. (2010). Data Mining: A Knowledge Discovery Approach 1st . Springer Publishing Company, Incorporated. ISBN: 1441941207 9781441941206.

[37] Hand, D., Mannila, H., and Smyth, P. 2001. *Principles of data mining*. Principles of data mining /David Hand, Heikki Mannila and Padhraic Smyth. Cambridge, Mass.: The MIT Press, c2001. (Adaptive computation and machine learning).

[38] Klaus, J. and Marc, D. (2002). Mining intrusion detection alarms for actionable knowledge. Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, Edomonton, Alberta, Canada.

[39] Mamdouh, R., Markus, S., Toby, J., and Ian, H. (2009). Data Mining Techniques and Algorithms. Prentice Hall, USA.

[40] Han Jiawei and Kamber Micheline. (2006). Data Mining: concepts and Techniques, 2nd edition, Morgan Kaufman.

[41] Agrawal, R., Imieliński, T., and Swami, A. (1993). "Mining association rules between sets of items in large databases". *Proceedings of the 1993 ACM SIGMOD international conference on Management of data - SIGMOD '93*. p. 207. ISBN 0897915925. Doi:10.1145/170035.170072

[42] Sergey, B., Rajeev, M., Ullman, J. D., and Tsur, S. (1997). Dynamic Itemset Counting and Implication Rules for Market Basket Data. In Proceedings of ACM SIGMOD Intl. Conf. On Management of Data (SICMOD'97), 25-264, Tucson, Arizona, USA, May 1997.

[43] Zaki, M. 2000. "Scalable algorithms for association mining", *IEEE Transactions on Knowledge and Data Engineering*, 12(3), (2000), 372-390.

[44] Han, J., Pei, J., and Yin, Y. Mining frequent patterns without candidate generation. In W. Chen, J. F. Naughton, and P. A. Bernstein, editors, *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, pages 1–12. ACM Press, 2000.

[45] Goswami, D. N. and Anshu C. (2010). An algorithm for frequent mining based on apriori, In (IJCSE) *International Journal on Computer Science and Engineering*, 2, 942-947.

[46] Srikant, R. (1996). Fast algorithms for mining association rules and sequential patterns. PhD thesis, University of Wisconsin, Madison.

[47] Vaarandi, R. and Pondins K. 2010. Network IDS alert Classification with frequent itemset mining and data clustering, Network and Service Management (CNSM), 2010 International Conference, 451-456, 25-29 Oct. 2010 doi:10.1109/ CNSM.2010.5691262.

[48] Wang, M.-F., Wu, Y.-C., and Tsai, M.-F. “Exploiting Frequent Episodes in Weighted Suffix Tree to Improve Intrusion Detection System,” 22nd International Conference on Advanced Information Networking and Applications - Workshops (aina workshops 2008) IEEE, pp. 1246–1252, Mar. 2008.

[49] Lippmann, R. J. W. and Haines et al. 2000a. The 1999 DRPA off-line intrusion detection evaluation", *Computer Networks-the International Journal of Computer and Telecommunications Networking*, 34(4), 579-595.

[50] Goeschel, K. 2016. Reducing false positives in intrusion detection systems using data-mining techniques utilizing support vector machines, decision trees, and naive Bayes for off-line analysis, proceeding., 2016 IEEE conference Norfolk, VA, USA, 10.1109/SECON.2016.7506774

[51] Alharby and Imai, H. 2005. IDS False alarm reduction using continuous and discontinuous patterns. Proceeding of ACNS 2005, Springer, Heidelberg, 192-205.

[52] Autrel, F. and Cuppens. 2005. Using an intrusion detection alert similarity operator to aggregate and fuse alerts. The 4th Conference on Security and Network Architecture Bat sur Mer, France.

[53] Jan, N., Shun-Chieh, L., Shian-Shyong, T., Nancy P., and Lin, A. 2009. Decision support system for constructing an alert classification model. *Expert Systems with Applications*, 36, 11145-11155.

[54] Viinikka, J., Debar, H., M'e, L., Lehikoinen, A., and Tarvainen, M. 2009. Processing intrusion detection alert aggregates with time series modeling. *Information Fusion*, 10, 312-324.

[55] Ghorbani, A., Lu, W., and Tavallaee, M. (2010). Alert Management and Correlation, Book Chapter, Book Title: Network Intrusion Detection and Prevention, Springer Us, 47, 129-160.

[56] Alshammari, R., Sonamthiang, S., Teimouri, M., and Riordan, D. (2007). Using Neuro-Fuzzy Approach to Reduce False Positive Alerts, Communication Networks and Services Research, 2007. CNSR '07. Fifth Annual Conference on, 345-349. Doi:10.1109/CNSR.2007.70.

[57] Debar and Wespi, H. A. 2001. Aggregation and Correlation of Intrusion-Detection Alerts. Recent Advances in Intrusion Detection.

[58] Kruegel, C. and Robertson W. (2004). Alert Verification: Determining the Success of Intrusion Attempts. In PROC. First workshop the detection of intrusions and malware and vulnerability assessment.

[59] Hachmi, F. and Limam, M. "Two-stage techniques to improve intrusion detection systems based on data mining algorithms". **Published in:** Modeling, Simulation and Applied Optimization (ICMSAO), 2013 5th International Conference on 2013. **ISBN:** 978-1-4673-5814-9 published by IEEE.

[60] Al-Mamory, S. O. and Zhang, H. (2010). New data mining technique to enhance IDS alarms quality, Journal in computer virology, 6(1), 43-55. Doi:10.1007/s11416-008-0104-2.

[61] Subbulakshmi, T., George, M., and Mercy, S. (2010). Real time classification and clustering of IDS alerts using machine learning algorithms. *International Journal of Artificial Intelligence & Applications*, 1, 1-9.

[62] Christopher, K., Fredrik, V., and Govamni, V. (2005). *Intrusion Detection and correlation: Challenges and solutions*, Springer.

[63] Nien-Yi Jan, Shun-Chieh Lin, Shian-Shyong Tseng, and Lin, N. P. 2009. A decision support system for constructing an alert classification model. *Proceeding Expert Systems with Applications*, 36 (2009) 11145-11155.

[64] Khanchi, S. and Adibnia, F. 2002. False alert reduction on network-based intrusion detection system by means of feature frequencies. Advances in Computing, Control, & Telecommunication Technologies, 2009. ACT '09. International conference on vol., no., 513,516,28-29 Dec.2009. doi: 10.1109/ACT. 2009.221.

[65] Weinstock, C. B. and Goodenough, J. B. (2006). On systems scalability. Software Engineering Institute, Technical Report, SEI, Carnegie Mellon University, CMU/SEI-2006-TN-012.

[66] Mamdouh, R., Markus, S., Toby, J., and Ian, H. 2009. *Data Mining Techniques and Algorithms*. Prentice Hall, USA.

[67] Anderberg, M. R., Cluster Analysis for Applications, Academic Press, New York, 1973, 162-163.

[68] Costa, Luciano da Fontoura and Cesar, R.M. 2001. Shape Analysis and Classification, Theory and Practice, CRC Press, Boca Raton, 577-615.

[69] Veyssieres, M. P. and Plant, R. E. 1998. Identification of vegetation state and transition domains in California's hardwood rangelands. University of California.

[70] Rokach, L. and Maimon, O. 2005. Data Mining and Knowledge Discovery Handbook, Tell-Aviv University, 321-349.SPIN 11053125,11411963.

[71] Teknomo, K. 2007. K-Means Clustering Tutorials. is available on http:\\people.revoledu .com\kardi\ tutorial\kMean\.

[72] Mohiuddin A. and Mohmood, A. N. 2014. "Network Traffic Analysis based on Collective Anomaly Detection" 9th Conference on Industrial Electronics and Application ICIEA, IEEE,PIN: 978-1-4799-4315-9/14.

[73] AlSaide, Almomani, Sathiyamoorthy, and Manickam, 2012 "Collection Mechanism and Reduction of IDS Alert" proceeding of the International Journal of Computer Applications. DOI: 10.5120/9274-3530, November, 2012.

[74] Schmidt-Thieme, L. 2004. "Algorithmic Features of Eclat" Conference: FIMI'04, Proceedings of the IEEE ICDM Workshop on Frequent Itemsets Mining Implementation, Brighton, UK, November 1.