

**IDENTIFYING MALICIOUS ACTIVITIES IN HONEYNETS  
USING CLUSTERING**

BY

**MUHAMMAD SHOIEB ARSHAD**

A Thesis Presented to the  
DEANSHIP OF GRADUATE STUDIES

**KING FAHD UNIVERSITY OF PETROLEUM & MINERALS**

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the  
Requirements for the Degree of

**MASTER OF SCIENCE**

In

**COMPUTER NETWORKS**

MAY 2012

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS DHAHRAN  
31261, SAUDI ARABIA

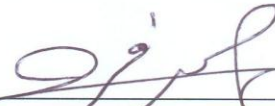
DEANSHIP OF GRADUATE STUDIES

This thesis, written by Muhammad Shoieb Arshad under the direction of his thesis advisor and approved by his thesis committee, has been presented to and accepted by Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE IN COMPUTER NETWORKS**.

Thesis Committee



Dr. Mohammed H. Sqalli(Advisor)



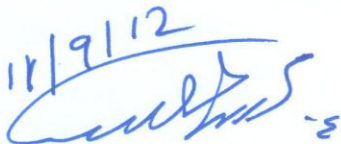
Dr. Farag Azzedin (Member)



Dr. Zubair Ahmed Baig (Member)



Dr. Basem AL-Madani  
(Department Chairman)

11/9/12  


Dr. Salam Adel Zummo  
(Dean of Graduate Studies)



**Dedicated**

to

*My Parents*

&

*My Beloved family*

# Acknowledgements

*In the Name of Allah, the Most Beneficent, the Most Merciful.*

Praise belongs to Allah, the Lord of all the worlds (2) The All-Merciful, the Very-Merciful. (3) The Master of the Day of Requital. (4) You alone do we worship, and from You alone do we seek help. (5) Take us on the straight path (6) The path of those on whom You have bestowed Your Grace, Not of those who have incurred Your wrath, nor of those who have gone astray. (7)

Al-Fatiha

I begin with the name of Allah, the most beneficent, the most merciful. May Allah bestow peace on our beloved Prophet Mohammed (*peace and blessings of Allah be upon him*), and his family. I would not have able to complete this work without the help of Allah who endowed me with health, courage, aptitude and patience.

During this work my parents and my family were a constant source of motivation and support. Their prayers, love and encouragement helped me to arrive at this milestone

Acknowledgements are due to *King Fahd University of Petroleum and Minerals* which gave me the opportunity to pursue a graduate degree and also for all the support I received in carrying out this research. I am also grateful to the *Saudi HoneyNet Project* at *KFUPM* for its support during this research.

I would like to express my gratitude to my thesis advisor Dr. Mohammed Houssaini Sqalli for all the things he taught me, for his patience when I couldn't get things done and for his help when I needed it. I am also very thankful to my thesis committee members Dr. Zubair Baig and Dr Farag Azzedin for their involvement and encouragement.

# Table of Contents

List of Figures .....	viii
List of Tables .....	x
Thesis Abstract.....	xii
ملخص الرسالة.....	xiii
CHAPTER 1 Introduction.....	1
1.1 Honeypots .....	2
1.2 Data Mining and Anomaly Detection.....	6
CHAPTER 2 Problem Statement.....	9
2.1 Background.....	9
2.2 Aspects of Anomaly Detection.....	11
2.2.1 Nature of Input Data .....	11
2.2.2 Types of Anomalies .....	12
2.2.3 Data Labels .....	13
2.2.4 Output .....	14
2.3 Motivation.....	15
2.4 Scope of work .....	17
CHAPTER 3 Literature Survey .....	18
3.1 Anomaly Detection Techniques.....	19

3.2 Clustering Techniques for Anomaly Detection .....	24
CHAPTER 4 Proposed Solution.....	31
4.1 Introduction.....	31
4.2 Clustering.....	33
4.3 Categorization of Clustering Methods .....	34
4.4 DBSCAN .....	37
4.5 Hierarchical Clustering .....	39
4.5.1 Agglomerative Clustering .....	40
4.5.2 Linkages.....	44
CHAPTER 5 DBSCAN .....	47
5.1 Implementation .....	47
5.1.1 Parameter Tuning.....	48
5.2 Experimental Results .....	51
5.2.1 Scan 14.....	53
5.2.2 Scan 19.....	55
5.2.3 Scan 28.....	58
5.3 Results and Discussion .....	63
CHAPTER 6 Hierarchical Clustering.....	66
6.1 Implementation .....	66
6.2 Initial Experiments.....	67

6.2.1 Average Linkage Clustering .....	68
6.2.2 Centroid Linkage Clustering.....	72
6.3 Result Discussion.....	76
CHAPTER 7 Performance Analysis.....	77
7.1 Experimental Setup.....	77
7.2 Description of traces used.....	78
7.2.1 Scan 27.....	78
7.2.2 Dionaea Capture Trace .....	79
7.2.3 Lab Trace .....	80
7.3 Collected Results .....	83
7.3.1 Scan 27.....	84
7.3.2 Dionaea Capture Trace-1 .....	91
7.3.3 Dionaea Capture Trace-2.....	96
7.3.4 Lab Capture.....	103
7.4 Results Overview .....	108
CHAPTER 8 Conclusion and Future Work.....	111
8.1 Future Work .....	113
8.2 Limitations .....	114
References.....	115
VITAE.....	118

## List of Figures

Figure 1-1 Diagram showing an integrated honeypot configuration .....	3
Figure 1-2 High Interaction Honeypot working inside a DM.....	3
Figure1-3 Different tools used by the Honeynet Community [4].....	5
Figure1-4 Clusters (A and B) and Outliers ( $O_1$ .....	8
Figure 3-1 Two different attacks observed on two different sensors showing same time series [21].....	26
Figure 4-1 (a) Existing work by Sqalli et al. [19] (b) Our Proposed Solution.....	32
Figure 4-2 (a) p is directly density reachable from q (b) p is density reachable from q.	38
Figure 4-3 Dendrogram of Scan-14 .....	41
Figure 4-4 Non-monotonic dendrogram of scan-14 .....	42
Figure 4-5 Linkages [32] .....	46
Figure 5-1 DBSCAN Program interface developed in Visual Studio .....	48
Figure 5-2 KNN Plot for Scan-28 in Descending Order.....	50
Figure 5-3 Scan-14 3-D Clusters .....	55
Figure 5-4 Scan-19 3-D Clusters .....	57
Figure 5-5 Scan-28 Day-1 3-D Clusters .....	60
Figure 5-6 Scan-28 Day-3 3-D Clusters .....	62
Figure 6-1 Scan-14 Hierarchical clustering with average linkage.....	69
Figure 6-2 Scan-28 Day-3 Hierarchical clustering with average linkage.....	71
Figure 6-3 Scan-14 Centroid Linkage Clustering.....	73
Figure 6-4 Scan-28 Day-3 Centroid Linkage Clustering.....	75
Figure 7-1 3-D Hierarchical Clustering of scan 27.....	86



Figure 7-2 3-D DBSCAN clustering of scan 27 .....	88
Figure 7-3 3-D plot of the Hierarchical Clustering of Dionaea Capture Trace-1 .....	92
Figure 7-4 3-D plot of DBSCAN clustering of Dionaea Capture Trace-1 .....	94
Figure 7-5 3-D plot of the Hierarchical Clustering of Dionaea Capture Trace-2.....	98
Figure 7-6 3-D plot of DBSCAN clustering of Dionaea Capture Trace-2 .....	100
Figure 7-7 3-D plot of the Hierarchical Clustering for the Lab Capture Trace .....	104
Figure 7-8 3-D plot of DBSCAN clustering of Lab Capture Trace.....	106

## List of Tables

TABLE 5-1 Honeynet Traffic Test Datasets .....	52
Table 5-2 Scan-14 Anomalies detected by Sqalli et al. [19].....	54
Table 5-3 Scan -14 DBSCAN Clustering with Min-Max Values.....	54
Table 5-4 Scan-19 Anomalies detected by Sqalli et al. [19].....	56
Table 5-5 Scan -19 DBSCAN Clustering with Min-Max Values.....	56
Table 5-6 Scan-28 Day-1 Anomalies detected by Sqalli et al. [19].....	58
Table 5-7 Scan -28 Day-1 DBSCAN Clustering with Min-Max Values.....	59
Table 5-8 Scan-28 Day-1 Anomalies detected by Sqalli et al. [19].....	61
Table 5-9 Scan -28 Day-3 DBSCAN Clustering with Min-Max Values.....	62
Table 5-10 Result Comparison .....	65
Table 6-1 Scan-14 Clusters with min-max values .....	69
Table 6-2 Scan 28 Day-3 Clusters with min-max values .....	70
Table 6-3 Scan-14 Clusters with min-max values .....	73
Table 6-4 Scan-28 Day-3 Clusters with min-max values .....	74
Table 7-1 Scan-27 Dataset Details.....	78
Table 7-2 Dionaea Dataset Details .....	79
Table 7-3 Lab Trace Dataset Details .....	81
Table 7-4 Lab Trace Attacks Detail.....	82
Table 7-5 Reported Anomalies for Scan-27 .....	85
Table 7-6 Hierarchical Clustering of scan 27 with Min-Max Values.....	87
Table 7-7 DBSCAN Clustering of Scan 27 with Min-Max values .....	89
Table 7-8 Result Comparison for scan 27.....	90

Table 7-9 Reported Anomalies in Dionaea Capture Trace-1.....	91
Table 7-10 Hierarchical Clustering of Dioanaea Capture trace-1 with Min-Max Values.	93
Table 7-11 DBSCAN Clustering of dionaea capture trace -1 with Min-Max Values .....	95
Table 7-12 Comparison of the reported results with Clustering result.....	96
Table 7-13 Reported results for the Dionaea Capture trace-2 .....	97
Table 7-14 Hierarchical Clustering of Dionaea Capture trace-2 with Min-Max Values...	99
Table 7-15 DBSCAN Clustering of dionaea capture trace -2 with Min-Max Values .....	101
Table 7-16 Comparison of the reported results with Clustering result.....	102
Table 7-17 Reported results for the Lab Capture trace.....	103
Table 7-18 Hierarchical Clustering of Lab Capture trace with Min-Max Values .....	105
Table 7-19 DBSCAN Clustering of Lab Capture Trace with Min-Max Values .....	107
Table 7-20 Comparison of the reported results with Clustering result.....	108
Table 7-21 Performance of clustering algorithms against each Trace file .....	109
Table 7-22 Performance of clustering algorithms against each Anomaly Type.....	110

# Thesis Abstract

Name: Muhammad Shoieb Arshad

Title: IDENTIFYING MALICIOUS ACTIVITIES IN HONEYNET DATA USING CLUSTERING

Major Field: Computer Networks

Date of Degree: May, 2012

*Honeypots have cemented their place as a tool used by organizations to study and analyze the threats against their networks and to find the vulnerabilities within. The down side of using Honeypots is the extensive amount of data they produce, making it virtually impossible to analyze manually. Researchers have come up with different ways to identify malicious activities in the Honeypot data. In this thesis, we propose to use the clustering algorithms to improve on an existing entropy based scheme used for identifying malicious activities in Honeynet traffic. The existing scheme partially requires manual inspection of the output to identify the different malicious activities. In this work, we implemented two clustering algorithms namely Density Based Spatial Clustering of Applications with Noise (DBSCAN) and Hierarchical Clustering. Then, we applied these algorithms to datasets, i.e., PCAP traces, provided by the Honeynet organization. Our results were compared with those obtained by the earlier scheme, and they showed that the use of automatic clustering can produce similar results, as it was produced by manual inspection, with better time efficiency.*

## ملخص الرسالة

الاسم: محمد شعيب أرشد

عنوان الرسالة: تحديد الأنشطة الخبيثة في بيانات مصاد الشبكة باستخدام التقسيم

التخصص: شبكات الحاسوب

تأريخ التخرج: جمادى الآخرة 1433 هـ - (مايو 2012 م)

عززت مصاد الشبكات مكانها كأداة تستخدمها المنظمات لدراسة وتحليل التهديدات ضد شبكاتهم وللعثور على نقاط الضعف داخلها. إنّ الجانب السلبي لاستخدام مصاد الشبكات هو الكمية الكبيرة للبيانات التي تنتجها، مما يجعل من المستحيل تحليلها يدوياً. وقد توصل الباحثون إلى طرق مختلفة لتحديد الأنشطة الخبيثة الناتجة عن بيانات مصاد الشبكة. في هذه الرسالة، إننا نستخدم خوارزميات التقسيم للتحسين على النظام الحالي القائم على تحديد الأنشطة الخبيثة في بيانات مصاد الشبكة. إن النظام السابق يتطلب جزئياً المعاينة اليدوية للبيانات لتحديد مختلف الأنشطة الخبيثة. لذلك نفذنا خوارزمية التقسيم المكاني للتطبيقات المعتمدة على الكثافة مع الضوضاء والتقسيم الهرمي. ثم طبقنا هذه الخوارزميات على قواعد البيانات المقدمة من منظمة مصاد الشبكة. وتمت مقارنة نتائجنا مع نتائج النظام السابق حيث أظهرت أن استخدام التقسيم التلقائي يمكن أن يسفر عن نتائج مماثلة ومشابهة للنتائج المعاينة يدوياً مع كفاءة أفضل وقت.

شهادة ماجستير علوم

جامعة الملك فهد للبترول والمعادن

الظهران ، المملكة العربية السعودية

# **CHAPTER 1**

## **Introduction**

The Internet era has brought revolutionary changes to the way we communicate and exchange information. It has also increased our reliance on digital devices for communication, which makes us vulnerable to all the insecurities and threats of relying on the Internet. Decades ago, businesses use vaults and banks to hold their sensitive trade secrets and financial information. But now, a centralized data center holds all the sensitive and day-to-day information. This makes the network security an utmost priority for businesses. Any software used on the network has its own exploits and vulnerabilities, and hackers only need a single exploit to be able to access a system. And, they can even hide their tracks once they take control of the system. Honeypots are developed to track the activities of hackers and also to figure out which exploits hackers use to gain access of a system. Honeypots are useful tools but they may produce a very large amount of data, and analyzing this data becomes very difficult. In this work, we are developing a framework for automatic detection of anomalies through clustering techniques, from the data collected by Honeypots.

## **1.1 HONEYPOTS**

Lance Spitzner, introduced the concept of Honeynets and Honeypots [1]. A Honeypot is a computer software which is placed on the network with some intentional security holes. Honeypots do not have any production value to the network. The main task of the honeypot is to record every communication, to and from it, and mark it as suspicious. The key responsibilities of a Honeynet include data control, data capture, and data analysis [1]. A Honeynet is a network consisting of multiple Honeypots. A Honeynet is more complex in deployment and management, but it provides better and more reliable information about attacks. The idea behind deploying a Honeypot is that it will be targeted and compromised due to some network attack, and it will capture all the data packets sent and received during the attack. This will help to reconstruct the attack and understand the method and technique deployed by the attackers to compromise the system security. Figure 1-1 shows multiple Honeypots integrated as part of a network. Figure 1-2 shows the placement of a Honeypot inside the demilitarized zone (DMZ).

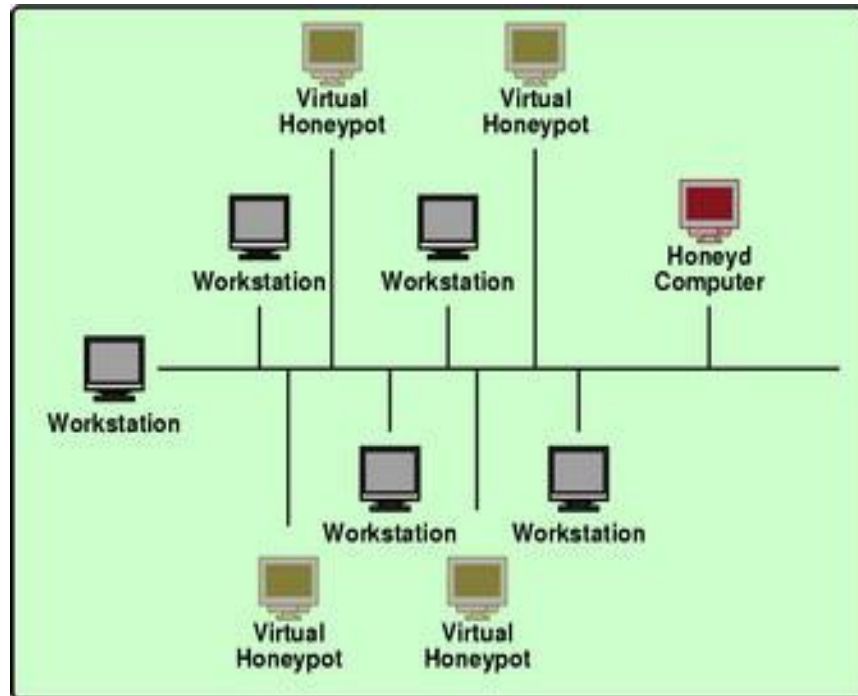


Figure 1-1 Diagram showing an integrated honeypot configuration

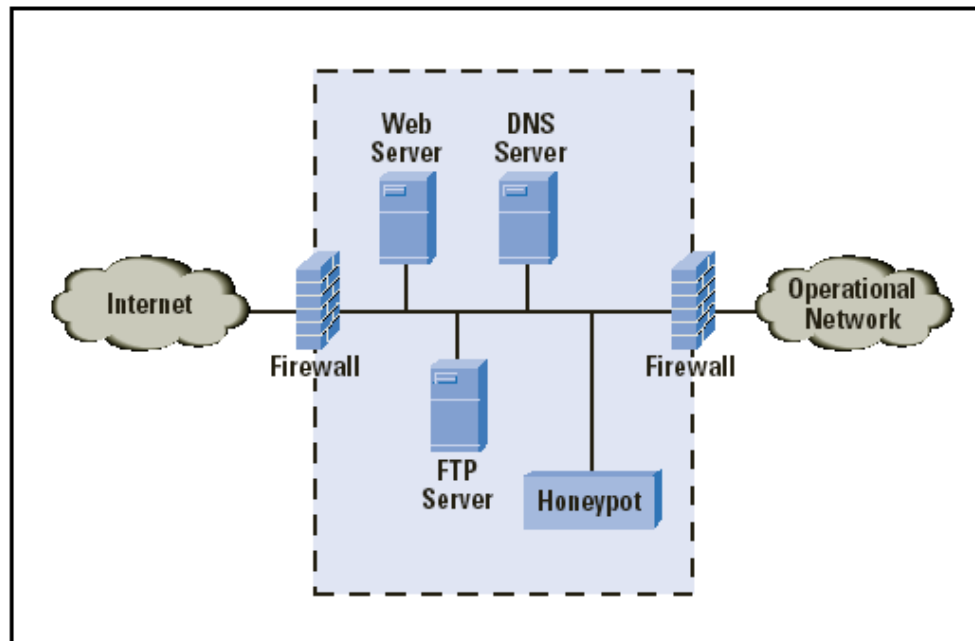


Figure 1-2 High Interaction Honeypot working inside a DM



Spitzner [2] introduced two types of Honeypots, High Interaction Honeypots and Low Interaction Honeypots.

- **High Interaction Honeypots** are real machines, same like other machines on the network, with all the software installed on them. And, they will even have some fake data. So, the hacker will not be able to identify whether it is a Honeypot or a real system. Figure 1-2 shows a high interaction Honeypot placed inside a DMZ.

- **Low interaction Honeypots**, on the other hand, are software with emulated services and very small interaction interface. Hackers can still exploit the vulnerabilities but they will not be able to control the machine after a successful attack.

There are currently different software packages available in the market which can provide the functionality of a Honeypot such as Nepenthes, Honeywall, HoneyD, Dionaea, etc. The difficulty associated with Honeypots is that they gather too much data and it can be of very different types, i.e., http, icmp, ftp, etc. Working with this extensive and varying types of data can create a lot of problems for the system analysts [3]. Figure1-3 shows the different tools available as honeypots.

# Tools Categories



Figure1-3 Different tools used by the Honeynet Community [4]

## 1.2 DATA MINING AND ANOMALY DETECTION

Data mining approaches applied to the network data can be classified into two different categories, misuse detection and anomaly detection. Misuse detection mainly focuses on detecting the known attacks and their variations. It tends to have very low false positive rate, mostly due to the attack signatures of the known attacks. On the other hand, anomaly detection techniques focus on the abnormalities in the network data which are deemed to be suspicious. These techniques can produce a large number of false positives due to the absence of any defined rules of abnormalities. However, anomaly detection has the ability to detect a zero-day attack, which is not possible with the misuse detection techniques. Misuse detection techniques are mostly used for real time analysis and detection by most of the antivirus tools and firewalls. On the other hand, anomaly detection techniques are usually used on the data collected during a network attack, but this detection is not achieved in real time. The reason behind anomaly detection techniques not being able to work in real time is that they usually need a certain time window to process the data and then identify its nature, whether it is normal or malicious. But in a real time environment, we cannot hold the data for the whole time window period, as it will greatly degrade the network performance.

Anomaly detection techniques are mostly based on either clustering or outlier detection. Clustering is the process of grouping data points together so that all data points with similar attributes can be grouped into the same cluster. Outlier detection is the process of identifying the data points which deviate considerably from regions of high density data points or other clusters. Figure1-4 shows clusters and outliers. Clustering and

outlier detection are both used for anomaly detection, but their use is mostly defined by the type of network traffic to which they are applied. If we want to apply anomaly detection techniques to normal network data, which also contains some intrusions, clustering will provide mostly the clusters of normal traffic or legitimate traffic, while outlier detection will provide the events which are not normal. And in this case, these outliers will be the network intrusions or some other type of attacks. So, if we want to apply the same techniques to the Honeynet data, then the term normal traffic will be redefined because all traffic coming to a Honeynet is considered suspicious. For Honeynet data, the majority of data does not represent normal or legitimate traffic. In the case of Honeynet traffic, clustering will provide us with different types of attacks; each attack is confined into a single cluster. Sometimes, instances of normal or legitimate traffic can also form a cluster, this mostly happens due to the presence of broadcast traffic in the network. But these clusters are easily identifiable due to their unique nature. In this case, an outlier can be one of two things, either an instance of normal traffic, or some attack which has not been seen before in the training set, including zero day attacks. To correctly identify the normal traffic, either in the form of cluster or outlier, we use known datasets for training purposes, so that the characteristics of the normal traffic can be identified correctly.

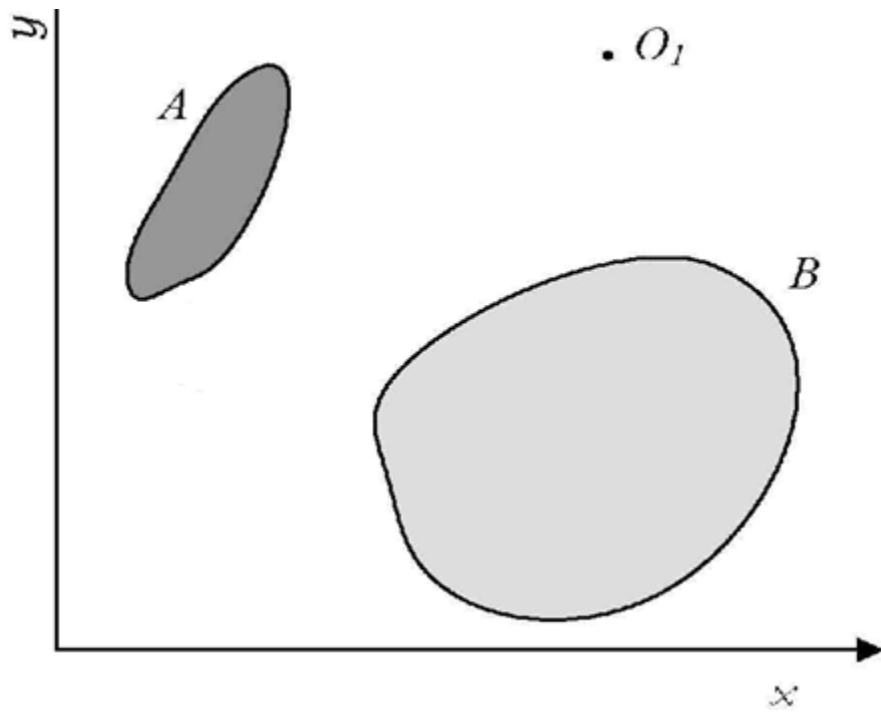


Figure1-4 Clusters (A and B) and Outliers ( $O_1$ )

## **CHAPTER 2**

### **Problem Statement**

#### **2.1 BACKGROUND**

Anomaly detection refers to finding data patterns that do not conform to the expected normal data behavior. These data patterns which do not conform to the expected normal data behavior are referred to as outliers, anomalies, peculiarities, etc.; but the names outlier and anomaly are more commonly used in the literature. The importance of anomaly detection techniques first came to light when these techniques were used in detecting frauds in health care, insurance, and credit card. Anomaly detection got more prominence most recently through their extensive use in the intrusion detection systems, military surveillance equipment, and fault detection in critical systems [5].

The main reason for the focus on the anomaly detection techniques is the fact that anomalies represent significant actionable information in the presence of very large datasets, i.e., data which needs human data analyzers attention. This happens in systems where anomaly detection is used to raise only alarms for anomalies but are not allowed to take actions. For instance, if applied to a credit card system, anomalies may lead to the detection of credit card misuse or identity theft. Also, anomalies in the bio-imagery of

human body may reveal some disease or tumor. And similarly, anomalies in the computer network may lead to the detection of already infected or hacked computer systems. Anomalies exist in a dataset due to a variety of reasons, but one thing which is common about anomalies is that they are useful for the data analyst. Their usefulness for the data analysts leads to the increased efficiency when dealing with very large datasets, as these anomalies point to the data analyst to where to look for the interesting information.

Anomaly detection is sometime compared with noise removal or noise accommodation [6]. Noise is the unwanted data which affects the data analysis process. Noise removal is the process of removing this unwanted data from a large dataset, while in noise accommodation we create a statistical estimation model to normalize the effect of noise in the dataset. Noise and anomaly are similar in a way that both deviate from the normal data, but noise is a hindrance to the existing data and it should be removed or accommodated. However, anomaly is part of the data which represents abnormality, but it should not be removed or accommodated as anomalies tend to lead to important events.

Another aspect of the anomaly detection is the novelty detection [7]. Novelty detection is the process of detecting previously unseen groups of anomalous events. Novelty detection techniques are used to detect those patterns which are unavailable in the training dataset. The purpose of the novelty detection is to update the database of known anomalies; therefore we do not have to go back to the training dataset whenever a new anomalous, previously unseen, pattern is observed in the dataset. The difference between anomaly detection and novelty detection is that anomaly detection only detects the anomalous events based on the database of known anomalies created during the

training phase. Novelty detection not only detects the known anomalies, but it also updates the database if it finds an unknown anomaly in the dataset.

## **2.2 ASPECTS OF ANOMALY DETECTION**

In this section, we will discuss the different aspects and challenges related to the anomaly detection process.

### **2.2.1 Nature of Input Data**

One of the significant aspects of anomaly detection is the nature of the input data. The input mostly consists of data instances (also known as observations, vector, point, record, sample, entity, etc.) [8]. Each data instance can be represented by using different attributes (also known as dimensions, variable, feature, characteristic, field, etc.). These attributes mostly represent the values in the binary or numerical form. Each data observation may consist of a single feature (uni-variate) or multiple features (multivariate). In the case of multivariate data features, these features can be of the same type or a combination of different types of data.

The nature of input data plays an important role in the selection of the anomaly detection technique, as each anomaly detection technique is designed for a certain type of input data [8]. Most of the anomaly detection techniques assume that all the features of a data instance are independent of each other, but in some techniques, we can combine some features for better detection. To combine different features, there should be some existing relationship between the data features. These data relations can include sequence data where all members can be arranged sequentially, graph data where each member can



be presented as vertices which are connected to each other through edges, or spatial data where data members are similar to each other, e.g., vehicular data.

In our work, we will be using network trace files in PCAP format as input to the anomaly detection system.

### 2.2.2 Types of Anomalies

An anomaly is referred to as a **point anomaly** if a single data instance can define the anomalous event independently [8]. This type of anomaly is mostly used when a single event of abnormality has a very high significance. The most common use of this type of anomaly detection is in fraud detection for credit cards, as any single transaction is important in detecting the anomalous credit card activity.

An anomaly is referred to as **contextual anomaly** if a certain data instance is anomalous in a specific context [8]. It is quite possible that the same data instance, but with a different context or behavior, is considered normal or non-anomalous. In these techniques, the anomalous behavior is determined based on the values of contextual and behavioral elements in that specific context. The most common use of contextual anomalies is in time-series data and spatial data. For example, a credit card transaction late in the night or early morning, even of a small amount, can be considered anomalous if it does not coincide with the normal behavior. But, a similar transaction in other times of the day can be considered normal.

An anomaly is referred to as **collective anomaly** if a collection of data instances is anomalous with respect to the complete dataset [8]. It is important to note that these data instances are not anomalous individually, but as a collection they are anomalous. For

instance, a single instance of zero values is not considered anomalous, but a collection of 4-5 zero values will raise an anomalous flag. The reason behind the collective anomaly is that if a single event differs from the normal behavior, we can consider this as a mistake by the user; but if there is a set of events which differ from the normal behavior, then it is considered as an anomalous event that needs further investigation. The techniques used for collective anomalies are more difficult to implement as compared to the other two types of anomalies discussed earlier. The difficulty is mostly due to checking the presence of a collection of points instead of a single point.

In our work, we will focus on collective anomaly detection, because all network attacks consist of a group of packets sent to the target machine.

### **2.2.3 Data Labels**

Data labeling is a process in which data instances are labeled as normal traffic or anomalous traffic. Labeling is mostly done by the human analysts in order to provide the labeled training dataset. But having a labeled training dataset does not provide the complete solution since sometime there are anomalies for which there is no labeled data for the training purpose. These anomalies are mostly dynamic in nature and their pattern is always changing, making it very difficult to label. The availability or unavailability of a labeled dataset determines the mode of operation for the anomaly detection techniques. The following three modes of operation are discussed in the literature.

The **Supervised anomaly detection** mode is based on the assumption that the labeled training dataset is available [8]. Techniques based on supervised anomaly detection create

a predictive model based on the training dataset; then they use this model to determine the anomalies in the unknown dataset.

The **Semi-Supervised anomaly detection** mode is based on the assumption that the training dataset only contains labeling for the normal traffic [8]. Techniques based on semi-supervised anomaly detection create a normal behavior profile from the training dataset. This normal behavior profile helps to determine the nature of any new data instance; either it will be considered normal if it matches the existing profile or it will be marked anomalous if it does not match any existing normal behavior.

The **Un-supervised anomaly detection** mode is based on the assumption that the training dataset is not labeled [8]. The only assumption made is that the number of instances of normal traffic will be much more than the anomalous traffic.

In our work, we will be using the supervised anomaly detection mode, in which we will initially train our technique on a training dataset. Once we have the threshold values for the training datasets, then we will apply these thresholds to the unknown datasets.

#### **2.2.4 Output**

The final step of any anomaly detection technique is to provide the output of the detection process. The following are two methods that are most commonly used in the literature to represent the output of an anomaly detection technique.

The **Score** based output techniques use an anomaly score of each data instance representing the degree of irregularity to the normal behavior. This will help the task of

the data analyst by first looking at the top anomalies, or by defining a cut-off point for the anomaly score to find the top list of anomalies.

The **Label** based output technique labels each data instance as normal or anomalous.

In our work, we will provide label based output, in which we will mark the clusters as different types of malicious activities.

## **2.3 MOTIVATION**

Anomaly detection is the process of detecting patterns of events which do not coincide with the normal behavior of the data. In the last decade, we have seen a dramatic increase in the amount of data collected for various purposes. Therefore, it becomes more important and challenging to find the abnormalities or unusual events in the collected data. These events can lead to the detection of some unwanted data, e.g., intrusion detection in network traffic, or of something very useful such as the discovery of a new star in astronomical data. In both cases, anomaly detection helps in gaining the in-depth knowledge about the system and the way abnormalities work within that system. The task of anomaly detection becomes more important when there is a certain action associated with the detection of unusual events. For instance, in the case of fraud detection in credit card, whenever there is a fraud detected, all concerned parties and law enforcement agencies are notified so that they can take an appropriate action.

An important challenge when developing an anomaly detection technique is the unavailability of the labeled data. Therefore, most of the time, we work in an unsupervised anomaly detection mode, in which we do not have the information about

the number of anomalies in the data, and we detect anomalies after they are classified by the clustering algorithm. For unsupervised anomaly detection mode, we provide the dataset to the clustering algorithm, which creates the clusters based on the similarities between the values of traffic features in the dataset. Once the clusters are created, we study them to identify the type of anomaly they are representing.

When we have to apply anomaly detection techniques to a dataset provided by any honeypot, the task becomes more challenging. Here, we not only have to look for the top anomalous events or top outliers as in the case of normal network traffic where we were only interested in the outliers, but also for certain common behaviors too. In normal traffic, common behaviors are considered legitimate as most of the traffic is legitimate. However, in case of honeypots, common behaviors are considered anomalous as most of the traffic in a honeypot is attack traffic. This is due to the fact that, in the case of a honeypot, all the traffic coming to it is considered suspicious. Therefore, when analyzing honeypot network traffic, we need to focus on both aspects of anomaly detection, clustering and outlier detection. Clustering is important for honeynet traffic as most of the data will be categorized into different clusters, and these clusters will help in identifying the different types of attacks. In clustering, the goal is always to define clusters in such a way that each cluster represents a specific set of events or in our case specific malicious activities, which share some similarities or represent similar kind of traffic. Ideally we are interested in creating clusters such that each cluster represents a specific type of attack. Hence, the task of any anomaly detection technique for honeypot data is to classify all the known attacks by comparing current values of the clusters to the threshold values from the known anomalies database.

## **2.4 SCOPE OF WORK**

The main objective of this thesis work is to analyze the network data produced by honeypots. As this research work is strictly focused on detecting anomalies in the honeynet data, our scope will be limited to the data collected by honeypots. And, another important assumption which we made for this research work is the presence of the malicious activities in the network traffic. Since we are using clustering techniques, we need at least one anomaly to build a cluster around it. All the network traffic input files which we used for this research work contain at least one malicious activity. In addition, our proposed technique will only be applicable for offline data analysis. Since we will be using five minute sliding time windows to calculate the entropy values for the identified traffic features, this scheme is inefficient for real time environment. This is acceptable for our work because honeypots are not developed for real time protection of the network. The responsibility of the honeypot is to study the nature of attacks and help to build counter measures for future attacks on the network.

## **CHAPTER 3**

### **Literature Survey**

This chapter contains the study conducted to understand the current state of research about the Honeypots, anomaly detection for the honeypot data, and the clustering schemes used to detect and identify the anomalies in the honeypot data. The first step in any Honeypot design is the deployment technique. The second step is the data collection phase, and finally the data analysis phase. The deployment and data collection phases are important as they pave the way for the most important phase, which is the data analysis phase. Hacker's activities can be tracked by using the raw data captured by the Honeypots. However, this task can be very time consuming if we do not use the tools designed for automated analysis. The biggest challenge for the data analyst is to deal with the data overload, mostly caused by the amount of data or sometimes by the varying types of data collected by honeynets [9]. Honeynets are known by researchers due to their ability to expose vulnerabilities in networks and systems by recording the attacks against them.

This chapter is divided into two sections; the first section contains the current state of Honeypots and detection of anomalous activities and the second section focuses on the data mining and clustering techniques used for honeypot data analysis.

### 3.1 ANOMALY DETECTION TECHNIQUES

Anomaly detection techniques are very different from the signature-based detection techniques. Signature-based detection techniques are mostly based on a pre-defined set of activities, while anomaly detection techniques check for any irregularities or dissimilarities in the data. Anomaly detection has played an important role in various sections of network security and intrusion detection. Anomaly detection techniques are very helpful in detecting different kinds of attacks like [10]:

- Different types of buffer overflow with shell code
- New attacks based on various exploits
- Variants of already known attacks

The current existing techniques can be categorized into two major types.

#### 1- Traffic feature based detection techniques [11] [12]

Traffic features based techniques are those techniques which use the IP header information to detect anomalies. The IP header information includes the Source IP, Destination IP, Source port, Destination port, and sequence number. An important performance constraint associated with the feature based detection techniques is that they require the header inspection to get the information needed. Header inspection can be a time consuming process, making traffic feature based techniques very difficult to use in a real-time situation.



## 2- Traffic volume based detection techniques [13-16]

Traffic volume based detection techniques are useful when there is a very high change in the volume of network traffic. This volume can be defined by the amount of data bytes sent or received, or it can be the number of packets sent and received in each direction. Volume based detection techniques are good in detecting Denial of Service (DoS) attacks and certain types of flooding attacks. But they cannot perform well against exploit or shell code based attacks.

Lakhina et al. [11] proposed an anomaly detection method in which they stated that anomalies in the network traffic can be detected by the distribution of traffic features such as IP addresses and ports. They stated that a wide range of anomalies can be tracked down by traffic feature distribution along with entropy. They experimented on the whole network traffic as it has different types of normal and abnormal traffic. The authors observed that in a wide set of data, classifying the nature of anomalies is a demanding task as they are constantly changing. As the anomalies are fluctuating on a regular basis, it is not appropriate to use pre-determined anomaly thresholds in a procedure for detecting anomalies. The distributional features of traffic like IP addresses and port numbers are very helpful in detecting numerous anomalies. The main difference between the work proposed by Lakhina et al. [11] and the work proposed by Dainotti et al. [14] and Haggerty et al. [15] is that Lakhina et al. introduced a method of using traffic features such as IP addresses and ports to track down the anomalies, in contrast to the other two methods [14] [15] which use traffic volume to track down the anomalies. They argued that there is a limited number of anomalies which cause a noticeable change in the

volume based features, but significantly large number of anomalies can be detected by using their proposed feature based anomaly detection technique. The authors adopted the following traffic features: Source and Destination IP addresses, Source port, and Destination port. To isolate the normal and abnormal anomalies, they follow the Principal Component Analysis (PCA), which uses dimensionality reduction.

Nychis et al. [12] proposed an anomaly detection technique by using entropy values. For anomaly detection, their main focus was on examining the performance by analyzing different traffic features and behavioral features distributions. The behavioral features contain the degree of distribution, measuring the number of individual Source and Destination IP addresses. They found, after conducting various experiments, that the distributions of Source IP, Destination IP, and port address all produce the same behavior towards the detection of known anomalies. Authors stated that behavioral and flow size distributions are not related to each other and their combination is very useful in detecting anomalies which are not detected by the address and port distributions. To detect the similarities between different feature pairs, the authors used entropy values to detect the relationship between different feature pairs. The authors concluded that the traffic feature distribution should not only be limited to port/address features, and that more features can be added to increase the detection results.

Kind et al. [17] presented a new theory regarding the feature based anomaly detection of Lakhina et al. [11]. They proposed a graphical method, i.e., histograms, to detect the anomalies. The authors constructed histogram patterns of different features. To track down the anomalies, they find different histogram patterns for different anomalies. The detection of anomalies is achieved in four levels: choosing features and creating

histograms, mapping histograms into a metric space, clustering and extracting patterns, and then identifying the anomalies. In this research, the authors employed different traffic features including port numbers, TCP flags, Source and Destination ports etc. The Principal Component Analysis (PCA) is employed for dimensionality reduction. Lakhina et al. [11] used PCA to distinguish between normal and abnormal traffic. This approach uses the histogram instead of entropy in order to track down the anomalies.

Ping and Abe [16] proposed an anomaly detection technique to detect Denial of Service attacks by using Packet size entropy. The authors stated that packets are assigned fixed sizes by default in numerous applications when dealing with initial data request and response messages. For example, the FTP application has by default reserved 40 bytes for acknowledgement and 1500 bytes for full packet data. The packets which are produced, even in the attack, are of the same size. From the normal non-attack traffic data, the threshold of the entropy is calculated and an entropy value is assigned to the threshold. Once they have calculated the threshold values for the normal traffic, the task is then to examine when entropy values exceed the threshold value, and this will be considered as an attack. This research work allows for tracking down short term as well as long term attacks. Hence, it is an advanced approach in detecting the anomalies as compared to the volume-based schemes.

Al-Haidari et al. [18] proposed an entropy-based solution to protect firewalls against DoS attacks. They used packet size entropy and compared it with pre-defined threshold values to determine the nature of incoming traffic. Based on these values, they were able to identify the DoS attacks targeting firewalls among the normal network traffic. Based on their experimentation, they showed that the entropy based scheme leads to high

performance improvements in firewalls by isolating the DoS attacks from the normal traffic. This also led to the increase of the throughput, the decrease in the delay, and the high availability of firewalls.

Sqalli et al. [19] developed a technique based on entropy and volume values of selected features for classifying malicious activities in Honeynet traffic. Initially, they experimented with different combinations of the proposed entropy-based features such as IP address and port number, as well as volume-based features such as number of packets and total number of bytes. Based on their results, they decided to use a combination of three entropy-based features and two volume-based features for effective and efficient anomaly detection. The three entropy-based features are:

- Destination Port Entropy
- Source Port Entropy
- Destination IP Entropy

And the two volume based features are:

- Total Payload Bytes
- Packet Count

Entropy can be described as a measure of uncertainty or randomness associated with a random variable, or in the case of Honeynets, it will be the randomness associated with a specific feature of the data entering or leaving the network. Entropy is mostly used to find the deviation in the data items, and these deviations can help in detecting anomalies. Entropy  $H(X)$  is defined by equation [3-1].

$$H(X) = - \sum_{i=1}^n P(x_i) \log P(x_i) \quad [3-1]$$

Where:

$$P(x_i) = \frac{\text{Number of packets with } x_i \text{ as traffic feature}}{\text{Total Number of packets}} \quad [3-2]$$

Sqalli et al. [19] used five minute sliding windows to calculate the entropy for the selected features. They also used the same window size to calculate the volume-based features. After studying the relationship of these features with the known malicious activities in the training dataset, they were able to identify different threshold values for different kind of attacks. Then, they applied these threshold values to other traffic traces provided by the HoneyNet community. The limitation of this technique is that once the entropy and volume values are computed for any honeyNet trace, thresholds have to be manually found against these measured entropy and volume values. The manual determination of thresholds increases the chances of accurately detecting an anomaly, but it also significantly decreases the efficiency of the technique. Our work is based on this technique, but it automates the detection and identification process.

### **3.2 CLUSTERING TECHNIQUES FOR ANOMALY DETECTION**

Barbar et al. [20] presented and implemented an Audit Data Analysis and Mining (ADAM) system. ADAM is based on anomaly detection in a large dataset. The authors' proposed approach is to use a combination of the association of rule mining and classification mining techniques. Initially, ADAM is applied to the training data to create a normal usage profile for the data during an attack free period. Then, the algorithm is applied to the known attacks dataset to create an association rule profile for the known attacks. ADAM is designed to work as a real time anomaly detection tool, but its training

phase is performed on offline known data. Then, it is placed online with the profiles of known attack patterns and the normal usage. ADAM will monitor the incoming data in the form of a sliding window to match the incoming data patterns to already known ones. Any unknown pattern will be marked as suspicious and will be separated along with all its related information for further analysis.

Thonnard et al. [21] applied data mining techniques to the honeypot data to develop attack signatures of distributed and polymorphic attacks. This work relies on a quality-based clustering technique specifically designed to identify the groups of similar attacks. The technique used is applied to the data collected by the honeynet, and depends on the “time series” clustering. The time series is a graphical plot of the time against the sum of Source count for any specific type of attack. The authors used a graph-based approach to define and formulate the problem. The Symbolic Aggregate Approximation (SAX) method is used to find the similarity distance for different time series graphs. SAX tends to approximate different time series by segmenting them into time intervals of equal size and summarizing each of these intervals by their mean values and then comparing these values against the known anomalies. Figure 3-1 presents two different attacks on separate ports having similar time series graphs.

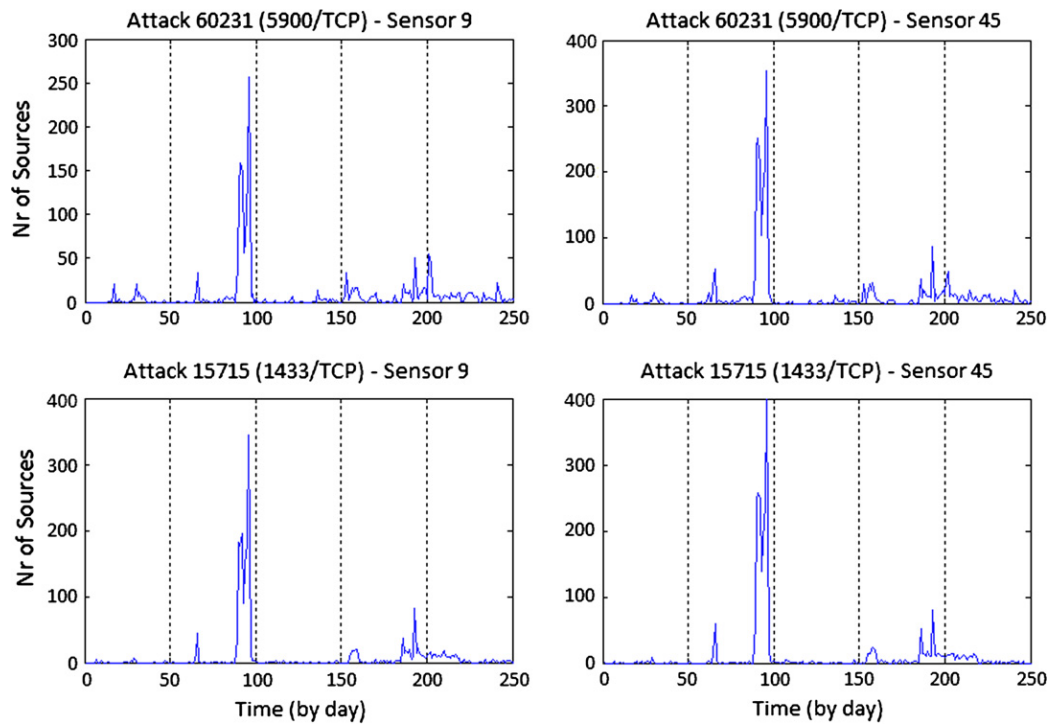


Figure 3-1 Two different attacks observed on two different sensors showing same time series [21]

Jin et al. [22] applied the knowledge discovery techniques on the data produced by a HoneyNet. They developed their own technique and named it K-Nearest neighbor Outlier Factor (KNOF) which is the combination of two other techniques K-Nearest Neighbor (KNN) [23] and Local Outlier Factor (LOF) [24]. The algorithm is divided into two major parts; the first part is visualizing the data, and the second part is identifying the outliers. For the visualization purpose, they used the Ordering Points to Identify the Clustering Structure (OPTICS) algorithm [25].

The working principal of OPTICS [25] is first to place data points into space, and these points can be based on multiple features, i.e., Source IP, Destination IP, port number, TTL, packet size, etc. Once all the data points have been placed into the space, then they will be categorized as a cluster or an outlier. To become a cluster, each point should have ' $k$ ' neighbors in its ' $\mathcal{E}$ ' neighborhood. ' $k$ ' is defined as the minimum number of data points inside the specified maximum radius ' $\mathcal{E}$ '. Data points outside the ' $\mathcal{E}$ ' neighborhood will be declared as outliers. This technique, OPTICS, will provide an augmented ordering of the data according to its density based clustering. These types of techniques are very helpful in visualizing the high-dimensional data. For the outlier detection, the authors used two different techniques, K-Nearest Neighbor (KNN) and Local Outlier factor (LOF).

KNN is a distance based outlier detection technique, while the LOF is a density based outlier detection technique. KNN does not need to have any prior knowledge of the dataset, as it bases its calculation on the distance value of the  $k^{th}$  nearest neighbor with reference to an object in the dataset. Initially, it will create a table containing each object in the dataset along with its  $k$ -distance, distance of the  $k^{th}$  neighbor from that object.



Then, this table will be sorted in descending order according to the  $k$ -distance. The first  $n$  objects in the table will be declared as outliers, where  $k$  and  $n$  are numbers defined by the user. The logic behind this assumption is the fact that sparse objects in the dataset have higher values of the  $k$ -distance than the objects in the dense neighborhood. As KNN takes the global view of the dataset to calculate the outliers, these outliers can be named as global outliers.

The Local Outlier Factor (LOF) of an object ' $p$ ' is the ratio between the average local reachability densities of  $p$ 's  $k$ -nearest neighbors to the local reachability density of  $p$ . The local reachability density for any object ' $p$ ' is the inverse of the average reachability distance based on the  $k$  nearest neighbors of ' $p$ '. The reachability distance between an object ' $p$ ' and any other object ' $o$ ' is the maximum of the  $k$ -distance of ' $o$ ' and the distance between ' $p$ ' and ' $o$ '. Based on these two techniques, the authors developed their own technique called  $k$ -Nearest Outlier Factor (KNOF) and it is defined for an object ' $p$ ' as the product of  $p$ 's LOF and the average reachability distance of its  $k$ -nearest neighbors. The reason to take this product is that the difference between LOF and the average reachability distance can be up to three to four times the value of LOF. So, to give proper representation to both protocols, the authors used the product of the two values instead of addition. When KNOF of all the objects have been calculated, a table will be created according to the descending value of KNOF, and the first  $n$  objects will be declared as outliers, i.e., malicious, in terms of intrusion detection.

Ghourabi et al. [26] presented a data analyzer for a honeypot router. A Honeypot router works as a honeypot, but with the added functionality of a router. To analyze the output of the honeypot router, the authors suggested a data mining based data analyzer

which will find the suspicious packets and will separate them for further analysis. Initially, they compared three different data mining approaches, namely Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [27], Cobweb [28], and  $k$ -mean [29]. They found that the DBSCAN approach has the minimum false positive rate, so they decided to use this scheme in their data analyzer.

Ghourabi et al. [26] derived their algorithm based on DBSCAN such that they will apply this approach to all the data and will create clusters according to this approach. All elements which are not classified as core objects or density reachable will be considered suspicious. And, they will be separated for human analysis. Each point represents a unique feature of the data, and in their case, the authors selected the following features: Source IP, Destination IP, protocol, TTL, packet length, and type.

Maheshwari et al. [30] discussed the limitations of the DBSCAN algorithm. The first limitation of DBSCAN is its inability to create clusters using time as a clustering parameter. Because the value of time increases linearly in the whole dataset, it is difficult to associate time with certain events and create clusters around it. The second limitation of DBSCAN is its inability to assign weights according to the density of the cluster. And it sometime fails to identify noise points between clusters of different densities. The next limitation is that DBSCAN clusters have different border densities inside a cluster. For very large databases, the memory requirement of DBSCAN becomes a problem as it has to process all the elements.

In this literature review, we thoroughly studied the current status of anomaly detection techniques[11, 16-19], and we also studied the developments made in the

clustering techniques [20-26]. This helped us in developing a framework which utilizes both the anomaly detection techniques discussed in [18] [19] and the clustering techniques discussed in [22] [26]. Our study of anomaly detection techniques [11] [16] [19] showed us that Entropy values of different traffic features are found to be very helpful in detecting the anomalies in the Honeynet data. While the study of clustering techniques discussed in [20] [21] [25] helped in designing the automatic detection framework. In this thesis, we will use Entropy based anomaly detection techniques along with the clustering algorithms to automate the anomaly detection process.

## **CHAPTER 4**

### **Proposed Solution**

#### **4.1 INTRODUCTION**

The idea of this thesis work is to automate the detection process of anomalies in the data collected by Honeypots. After studying different solutions, we are able to develop our own framework for the automatic detection of anomalies. In our framework we will take a dataset produced by a honeypot, and will use the selected traffic features for the calculation of entropy values. Once we have the entropy values, we will then use the selected features to create the clusters using the proposed clustering algorithms. After the clusters are created, the maximum and minimum values for each cluster will be compared with the detected values for the known anomalies. Then, cluster names will be assigned based on the similarities with the threshold limits of known anomalies. Figure 4-1 shows a comparison between the work of Sqalli et al. [19] and our proposed solution. We will experiment with density based clustering and hierarchical clustering as these two clustering schemes do not require the number of clusters as an input parameter. We do not want to fix the number of clusters in our research work as this will allow us to apply our technique to various types of network data where the output may not be a fixed

number of anomalies. In this way, we can have different number of clusters based on the number of anomalies inside the dataset.

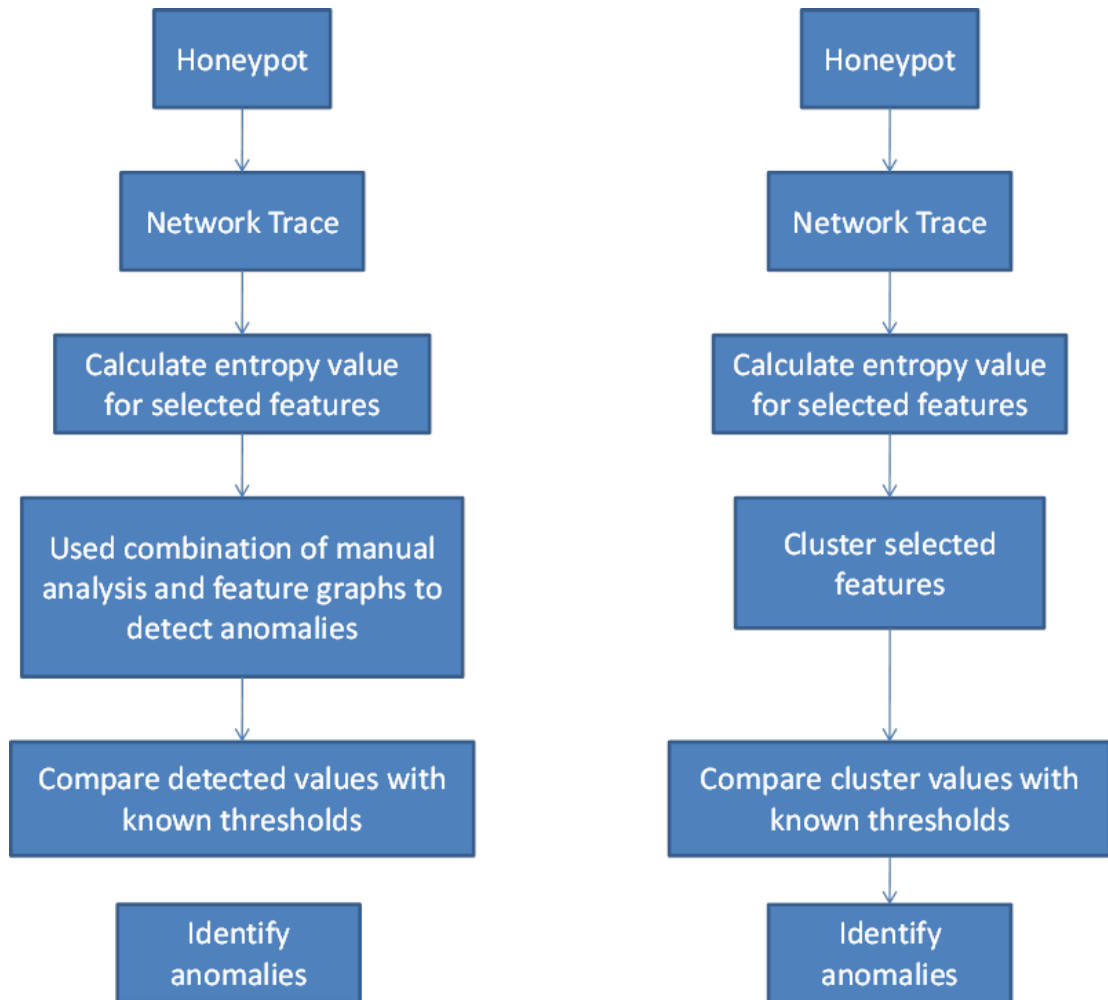


Figure 4-1 (a) Existing work by Sqalli et al. [19]

(b) Our Proposed Solution

## 4.2 CLUSTERING

Clustering is a process of grouping the data into similar groups or clusters, so that objects inside a cluster have high similarity in comparison to one another, but are very dissimilar to objects in other clusters. Dissimilarities are assessed based on the feature values describing the objects. Common features include distance, density, probability, etc. Clustering is also known as data segmentation in some applications, as it is used to partition large datasets according to certain similarity criteria. Clustering algorithms are used in various fields of science to manage large sets of data and information. There are some recommendations in the literature about choosing the best clustering algorithm for a particular dataset. The following are typical requirements of clustering in data mining [9]:

- **Scalability:** Ability to perform well when applied to large datasets. High scalability in clustering is needed.
- **Ability to deal with different types of attributes:** Data handling should not be limited to numerical data; an application may require the clustering algorithm to handle data other than numerical data.
- **Discovery of Clusters with arbitrary shape:** A clustering algorithm should be able to detect clusters not only in the standard geometrical shapes but also in arbitrary shapes.
- **Minimal requirements for domain knowledge to determine input parameters:** The ability of a clustering algorithm to start the clustering process without having too much information about the dataset is also needed. And the output produced by the clustering algorithm should not be over reliant on the

input parameters provided by the users, because these parameters are not very easy to determine, and they may lead to biased results. These parameters can have either positive or negative effect on the results; this effect depends on the precision of the initial parameters. The output of some clustering algorithms is largely affected by the preciseness of the input parameters. For those clustering algorithms, performance solely depends on the accuracy of the input parameters.

- **Ability to deal with noisy data:** A clustering algorithm's performance should not be affected by the presence of noise in the dataset.
- **Incremental clustering and insensitivity to the order of input records:** A clustering algorithm should be able to include new data to the existing clustering structure, instead of starting the whole clustering process again. And, the order of input data should not affect the outcome of the clustering process.
- **High dimensionality:** A clustering algorithm should be able to perform with the same efficiency for the datasets with higher number of dimensions as it performs for the datasets with lower number of dimensions.
- **Constraint-based clustering:** A clustering algorithm may also have to ability to add constraints before starting the clustering process. Different applications have different constraints for the clustering.

### 4.3 CATEGORIZATION OF CLUSTERING METHODS

There exist many clustering algorithms in the literature, but there is no fixed categorization of these clustering algorithms. Some clustering algorithms use multiple features to improve their output, which makes it difficult to categorize them into a single

category. In general, the major clustering methods can be classified into the following categories: [9]

- **Partitioning Method:** A partitioning method divides all the data points into  $k$  partitions, where  $k$  is a pre-defined number. Each partition is a cluster and satisfies the condition that each cluster contains at least one data point, and each data point should be part of only one cluster. The general criterion of a good partitioning is that objects in the same cluster are “close” or related to each other, whereas objects of different clusters are “far apart” or very different.
- **Hierarchical Method:** Hierarchical clustering creates clusters by arranging all the elements into a specific hierarchy. This hierarchy can be built from bottom-up, as in the case of agglomerative clustering, or from top-down, as in the case of divisive clustering. In agglomerative clustering, each data point is assigned to a separate cluster, and then it will start merging the clusters based on the similarities between each other. It will stop when there will be only one cluster or when it will reach a pre-defined stopping condition. In divisive clustering, all the data points are assigned to the single cluster in the beginning. Then, we start breaking the large cluster into smaller ones, which are similar. Divisive clustering also requires a predefined stopping condition to halt the clustering process.
- **Density-based Method:** The majority of clustering algorithms are based on the distance between the data points, but these clusters are not helpful in creating arbitrary shaped clusters. In density based clustering, clusters are



created based on the density of data points close to the starting data point. And the cluster shape follows the density pattern in the “neighborhood”, instead of following a standard geometric shape. “Neighborhood” is defined by the presence of a minimum number of data elements in the pre-defined radius. DBSCAN and its extension, OPTICS, are the most common density based clustering algorithms.

- **Grid-based Method:** In grid-based clustering, we first create a grid-like structure consisting of cells. And, each cell in the grid structure represents the data points from the given dataset. Each data point is the combination of all the selected features for that specific packet. The benefit of the grid-based clustering algorithms is the fast processing time, as these algorithms are independent of the number of data points. The grid-based clustering algorithms only depend on the number of cells in each dimension.
- **Model-based Method:** In model based clustering, we first create a statistical model for each known type of cluster. And when we apply these algorithms to the datasets, it starts matching each cluster model with the incoming data. Incoming data is assigned to the clusters based on their similarity with the existing model.

Out of these cluster categorizations, we have decided to use the density-based method and Hierarchical method. One reason for selecting density-based method and hierarchical method is that we do not know the exact number of anomalies or clusters in a given dataset. Secondly, the shape of the clusters formed can be arbitrary. In both density-based and hierarchical methods, we do not need to specify the number of clusters and these

methods can handle clusters with arbitrary shape. Therefore, density-based and hierarchical clustering methods are better suited for our requirements. In density based clustering, we are going to use the Density-Based Spatial Clustering of Applications with Noise (DBSCAN). In Hierarchical clustering, we are going to use the agglomerative (bottom-up) clustering.

#### 4.4 DBSCAN

The Density Based Spatial Clustering of Applications with Noise (DBSCAN) [27] is a Clustering based algorithm. The idea of DBSCAN is that each cluster must contain a predefined minimum number of data points in its neighborhood. The minimum number of neighbors and the neighborhood radius are both predefined. Epsilon ( $Eps$ ) is the maximum radius of the neighborhood. The maximum radius for any data point is defined as the maximum distance from that data point, considering it as the neighborhood center. The circular area surrounding a data point is called  $Eps$ -neighborhood for that data point. And  $MinPts$  is the minimum number of points in the  $Eps$ -neighborhood. The DBSCAN algorithm defines two sets of objects: core objects and density reachable. Core objects are those that contain  $MinPts$  in their  $Eps$ -neighborhood, and density reachable objects are those objects which exist inside an  $Eps$  of a core object, but do not have  $MinPts$  points. A data point can either be a single point, then it will be called directly density reachable as in Figure 4-2 (a), or it can be a chain of points such that each point is directly density reachable from another point, then the two points at the end of the chain are called density reachable as in Figure 4-2 (b).

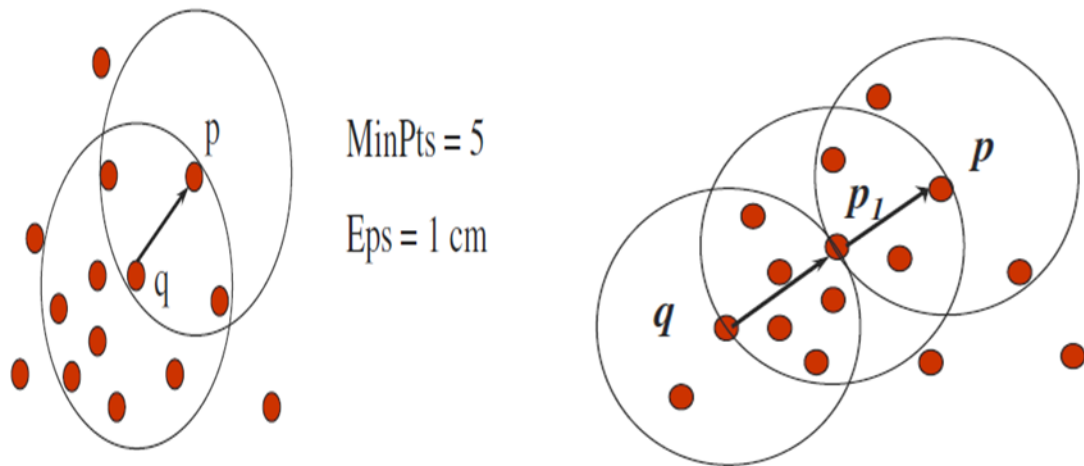


Figure 4-2 (a)  $p$  is directly density reachable from  $q$

(b)  $p$  is density reachable from  $q$  [8]

## 4.5 HIERARCHICAL CLUSTERING

Another type of clustering that is also present in the literature is called Hierarchical clustering. It provides a hierarchy or a structure as an output, which gives us more information about all the clusters. Hierarchical clustering also shows, in the form of a hierarchical tree view, the order in which different data points are merged from the initial to the last merge operation, while flat clustering cannot provide any information about merge operations performed during cluster formation. Algorithms used in Hierarchical clustering do not require the number of clusters as input, and most of these algorithms, which are also used in information retrieval, are deterministic. But, a drawback that is associated with the Hierarchical clustering is its inefficiency. This is due to the fact that Hierarchical clustering algorithms have quadratic complexity as compared to the linear complexity of the  $k$ -means or other flat clustering algorithms. Hierarchical clustering algorithms can be divided into two major categories [31]:

- 1- Bottom-Up Clustering (Agglomerative Clustering)
- 2- Top-Down Clustering (Divisive Clustering)

Agglomerative Clustering starts by assigning each data point into a separate cluster and then building up clusters from the bottom. On the other hand, Divisive Clustering starts by assigning all data points into a single cluster, and then breaking them into smaller clusters based on some defined criteria. In our research work, we have focused on the Agglomerative Hierarchical Clustering, as it is more flexible and it can operate with multiple clusters creation criteria. This flexibility of having multiple

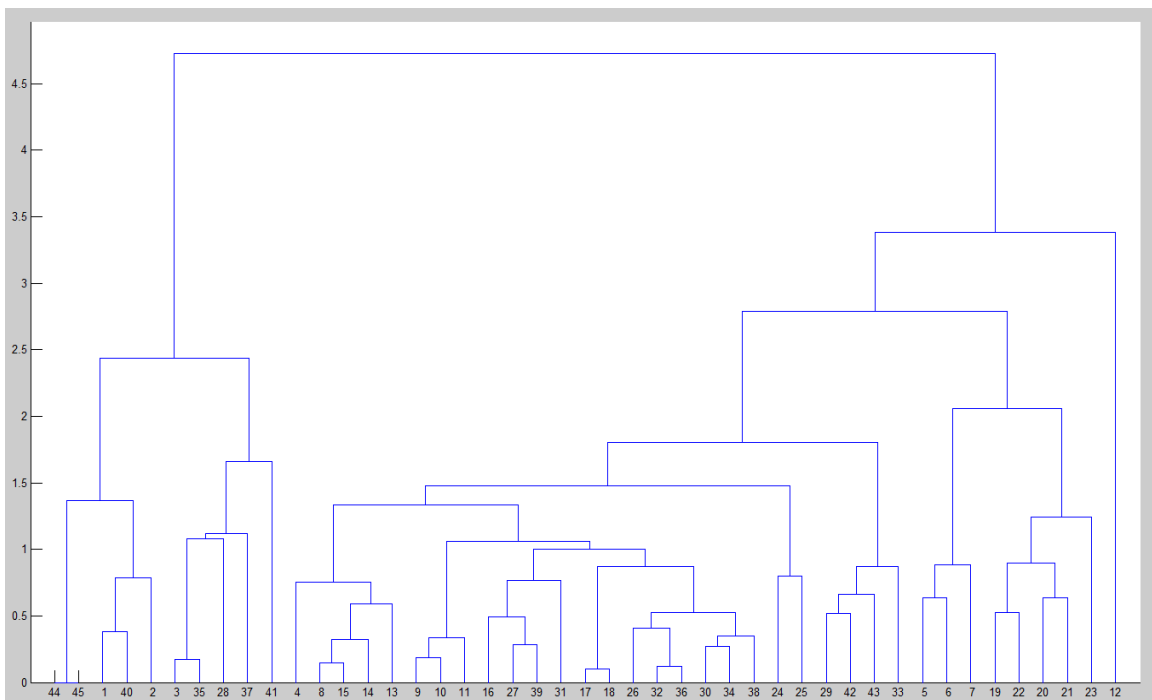
criteria will help us in finding the best criterion for anomaly detection in addition to allowing us to fine tune the performance of the detection algorithm.

#### **4.5.1 Agglomerative Clustering**

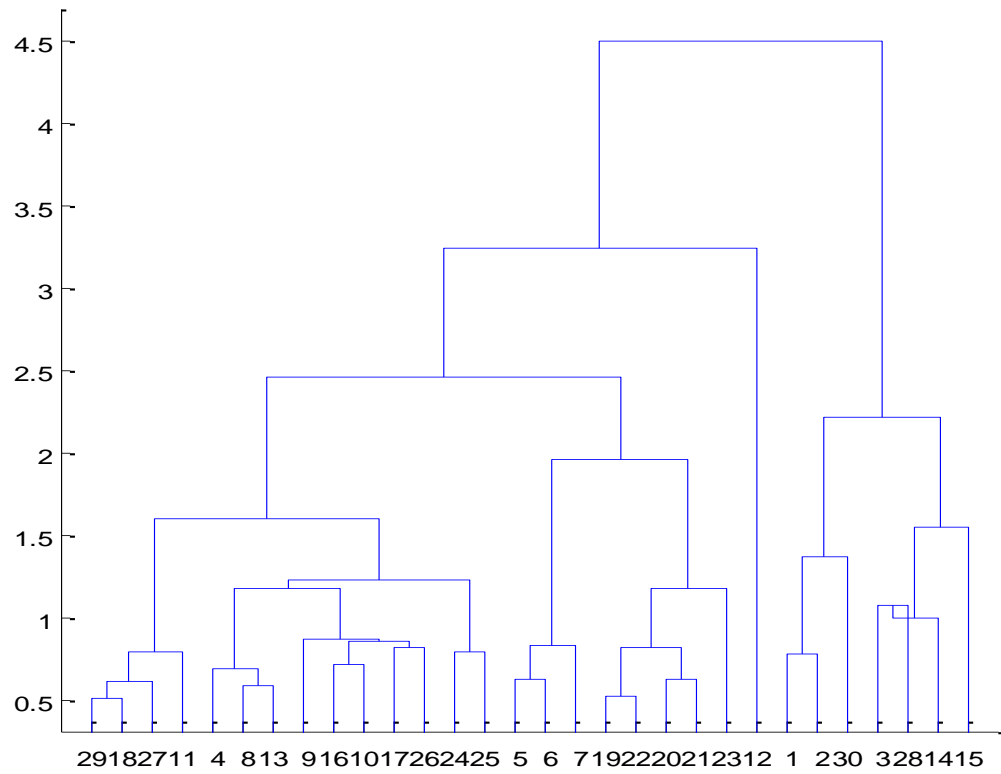
Agglomerative clustering is also known as the bottom-up clustering. In agglomerative clustering, initially each data point is assigned as a single cluster. Then, clusters which are similar to each other are merged. In every iteration, the clusters are merged based on the similarity between them. At the end of the clustering process, the hierarchy of the clusters can be viewed by a special type of graphs called “dendrogram”. In this graph, horizontal lines represent the merging of two clusters. The y-coordinate of these horizontal lines represent the value of similarity when they are merged, and this is called the combination similarity of the merged clusters. The similarity value depends on the criteria used to merge the clusters together. By going up in the dendrogram, we can reconstruct the history of all the mergers between the clusters. Figure 4-3 shows the dendrogram for the scan 14.

A fundamental assumption in agglomerative clustering is that all the merging operations are monotonic. Monotony in the merging operations means that the sum of the combination similarities of both data points merging to form a new point should be equal to or greater than the combination similarity of the new point created after the merging. If this condition is not met, then the merge operation will be called inversion. Inversion does not affect the performance or throughput of the clustering algorithm, but it is used to identify an abnormal merge operation. Usually, a centroid linkage has one or two inversions, and these inversions simply show the irregularities in a merge operation. The reason is that in a centroid linkage all merge operations are non-weighted merges. When

a large centroid, containing large number of data points, is merged with a very small centroid, containing very few data elements, then it is possible that the new centroid may have a combination similarity greater than the sum of individual similarities of the centroids merging to form a new centroid. Figure 4-4 shows a non-monotonic dendrogram of scan-14.



**Figure 4-3 Dendrogram of Scan-14**



**Figure 4-4 Non-monotonic dendrogram of scan-14**

The hierarchical cluster creation process has some similarities with the flat clustering, but it also offers options which are unique to the agglomerative clustering. As we have seen in Figure 4-3 , a dendrogram provides us with the hierarchy; but to create clusters, we need to define some cutoff criteria. The following three criteria are defined in the literature to create clusters [32]:

- The first criterion is called natural clustering. In this method, we look for the maximum difference between two combination similarities before any merge operation, and then we break these maximum difference merge operations to create clusters.
- In the second criterion, instead of going for the highest value of the difference between the individual similarities before the merge operation, we specify a certain value of the difference in individual similarity. Then, a cluster is created every time we encounter that the dissimilarity value is greater than the defined threshold.
- In the third criterion, as in the flat clustering, we provide the number of clusters required. This option of providing the number of clusters is optional and it can be used to further enhance the performance and throughput of the anomaly detection technique. Clusters are created at the top of the dendrogram by dividing the whole tree into the required number of clusters.



### 4.5.2 Linkages

Linkages play an important role in the agglomerative clustering. Linkages are the functions which are used to create the similarities or dissimilarities between the data points. In general, the role of the linkages is to define the criteria by which the small clusters will be merged. There are four different types of linkages defined in the literature [31].

- Single linkage
- Complete linkage
- Average linkage
- Centroid linkage

In the **single linkage** case, when we need to take a decision about merge operations, we only look at the value of the data points which are closest to each other. Or in simple words, we look for the closest link between two clusters, and this value for the closest link will be used to determine the possible merge operations. Decisions based on the single linkage are mostly considered as local decisions as they do not take the global picture into account.

In the **complete linkage** case, when we need to take a decision about the merge operations, we look at the value of the data points which are farthest from each other. Or in simple words, we look for the value of the farthest link between two clusters. This value will be used to determine the possible merge operations. The benefit of using the complete linkage is that the priority is given to the merge operations involving small clusters instead of merging big clusters first.

In the **average linkage** case, when we need to take a decision about the merge operations, we look at all the distance pairs in both clusters and take their average. Or simply, we can say that the average linkage is the average distance between all the data point pairs in both clusters. The average linkage helps us to overcome the shortcomings of the first two linkages as it does not only take into account the local perspective but the global one as well.

In the **centroid linkage** case, when we need to take a decision about the merge operation, we first calculate the centroid or center of mass, which is the average of all the data points inside the cluster. Once we have calculated the centroids, then we will start merging those centroids that are closest to each other. Closeness is decided by the distance between the centroids. In contrast to the other three types, the centroid linkage is sometimes non-monotonic or it has inversion in its dendrogram. This non-monotonicity is due to the recalculation of a centroid before every merge operation. As centroids are not actual data points, and needed to be calculated every time, this sometime leads to an increase in the distance similarity, which shows as an inversion in the dendrogram. Figure 4-5 shows a graphical presentation of the linkages discussed above.

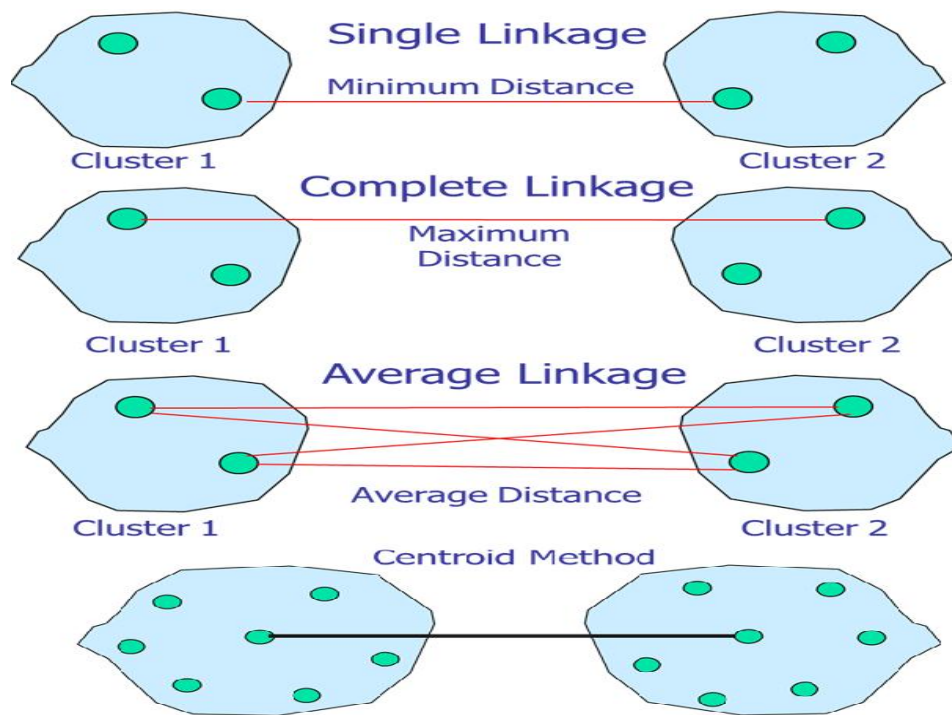


Figure 4-5 Linkages [32]

## **CHAPTER 5**

### **DBSCAN**

Density Based Spatial Clustering of Applications with Noise (DBSCAN) [27] is a density based clustering algorithm which we are going to use for this thesis work. Details of DBSCAN have been discussed in detail in Chapter 4. In this chapter we are going to implement DBSCAN and then we will tune the input parameters for the DBSCAN and finally we will test the performance of DBSCAN on different network trace files.

#### **5.1 IMPLEMENTATION**

Our aim in this work is to improve the efficiency of the earlier scheme by Sqalli et al. [19] by the automatic creation of clusters. This will remove the need of manually identifying the malicious activities inside the Honeynet traffic. For this purpose, we use a data mining based clustering algorithm namely Density Based Spatial Clustering of Applications with Noise (DBSCAN) [27].

For implementation purposes, we use Microsoft Visual Studio 2010 to develop our DBSCAN program. This program takes a comma separated file as input, which contains five columns. Each column represents the entropy values of the selected features, i.e., Source port, and Destination port, Source IP, as well as the values of the packet count,

and total payload bytes. In addition to the input file, the program requires two more input parameters,  $MinPts$  and  $\mathcal{E}$  ( $Eps$ ). Based on the value of  $MinPts$  and  $\mathcal{E}$  ( $Eps$ ), our program will process the input data to create the clusters. And, it will provide the output in the form of *clusters* and *noise*. “*Clusters*” will contain the data elements belonging to them, and “*noise*” will contain the data elements which do not belong to any cluster. Microsoft Visual Studio does not support the 3-D scatter plots. So, for the purpose of plotting graphs and creating plots, we have used MATALB 2011R2. Figure 5-1 shows a snapshot of the DBSCAN program interface developed in Microsoft Visual Studio 2010.

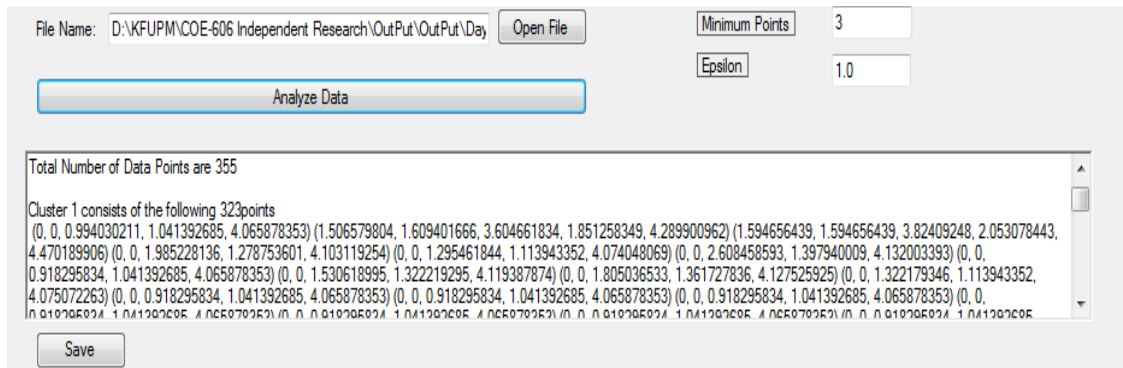


Figure 5-1 DBSCAN Program interface developed in Visual Studio

### 5.1.1 Parameter Tuning

Choosing the best value of “ $MinPts$ ” depends on the level of granularity required and the minimum number of points that should be available in each cluster. The maximum value of “ $MinPts$ ” depends on the total number of data elements in the dataset. A suitable value of “ $Eps$ ” will be the last dip or “valley” in the plot for any specific value of

“*MinPts*”, after which the plot line becomes almost a straight line [27]. Figure 5-2 shows the K-NN plot for the training dataset, i.e., Scan 28 Honeynet trace, with “*MinPts*”=2, 3, 4, 5, 6. A suitable value of “*Eps*” can be found for each value of “*MinPts*”. . In Figure 5-2, if we choose the value of “*MinPts*” to be 2, then by following the blue line we can find the last dip in the curve, which is around 0.70 for *Eps*. Sqalli et al. [19] used entropy values for three traffic features, and all of these values are in the range of zero to 20. Zero shows that there is no change in entropy and 20 shows the maximum change. However, the values of the two volume-based features were used by Sqalli et al. [19] rather than their entropy value, and the range for these values was much higher, e.g., 1 KB to 500 KB for the Total Byte feature. The magnitude of these values complicated the cluster creation process as they are larger by orders of magnitude compared to the other three entropy-based values. Therefore, we normalize the two volume-based features by taking the logarithm (log) of their respective values. This way, we can easily create a 5-dimensional space and apply DBSCAN on all five features for creating clusters. Then, we analyze the output to find how many clusters exist, which represent the malicious activities, and compare the output to the results obtained by Sqalli et al. [19]. In summary, the five dimensions used are: the Destination Port Entropy, the Source Port Entropy, the Destination IP Entropy, the Log of the Total Payload Bytes, and the Log of the Packet Count.

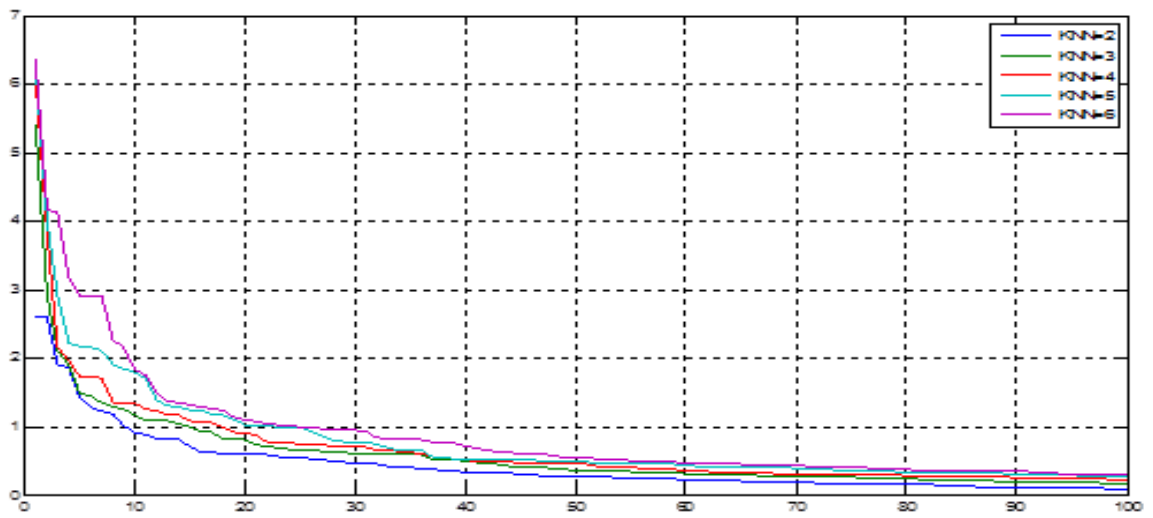


Figure 5-2 KNN Plot for Scan-28 in Descending Order

## 5.2 EXPERIMENTAL RESULTS

We implemented our approach so that it can take the five parameters as input and then create a 5-dimensional space to apply the DBSCAN algorithm. We adjusted the values of “*Eps*” and “*MinPts*” to fine tune the cluster creation process. Finally, we obtain the clusters along with the data points falling into each cluster. Then, we plot these clusters in a 3-dimensional space, because of the limitation in plotting higher dimensions. For the plotting purposes, we used the following three dimensions: Destination Port Entropy, Source Port Entropy, and Destination IP Entropy. All clusters shown in the 3-D plots are created using all five parameters.

For experimentation purposes, we used the same “Scan of the Month” traces which were used by Sqalli et al. [19]. These traces are taken from the “Scan of the Month” challenges available on the HoneyNet Organization’s public website [33]. TABLE 5-1 provides more detailed information about these traces.



**TABLE 5-1 Honeynet Traffic Test Datasets**

<b>Traffic Dataset Name &amp; Source</b>	<b>Traffic Details</b>	<b>Description</b>
Scan 28 - Honeynet.org – Scan of the Month	Day1: 18843 Packets – 24 Hours Day 3: 123123 Packets – 24 Hours	Trace collected by the Mexico Honeynet Team, Italian blackhats break into a Solaris server then enable IPv6 tunneling for communications.
Scan 14 - Honeynet.org – Scan of the Month	6707 packets Total Duration 20 Hours	This trace is about a successful Windows NT attack.
Scan 19 - Honeynet.org – Scan of the Month	24440 packets Total Duration 23 Hours	Trace of Redhat Linux 6.2 honeypot compromise.

### 5.2.1 Scan 14

Scan 14 is the first Honeynet trace used by Sqalli et al. [19], to identify malicious activities within, and 3 anomalies have been reported in this trace. When we run the DBSCAN clustering algorithm on the five output parameters, we obtained five clusters. Figure 5-3 shows the graphical output of the clusters for this trace. Out of these five clusters, Clusters 3, 4, and 5 exactly match the threshold of malicious activities found in [19]. Clusters 1 and 2 do not match any anomalous pattern defined in [19]. As explained earlier, clustering algorithms create clusters based on similarities between the selected traffic features. But, it is not necessary that all the clusters created will match the threshold values for the known attacks. The number of clusters produced is related to the level of granularity which we want from the data, so sometimes it will produce clusters which will not match any threshold of the known attacks. The tuning phase helps us identify the suitable values which will produce better detection and results. Each cluster is defined based on specific entropy and volume values for each traffic feature. Clusters 1 and 2, which do not represent any reported malicious activity, can either be considered as noise or as a new attack for which there is no threshold available. If we use a switch level trace file, it always contains traffic to or from other hosts which are connected to the same switch along with the Honeypot; and in this case, we can safely assume that these clusters represent noise. However, if it is a Honeypot level trace file then we have to further investigate the cluster. Table 5-2 presents the anomalies detected by Sqalli et al. [19]. Table 5-3 presents the DBSCAN Clustering results with the minimum and maximum values for each traffic feature. Each cluster present in these tables is shown as

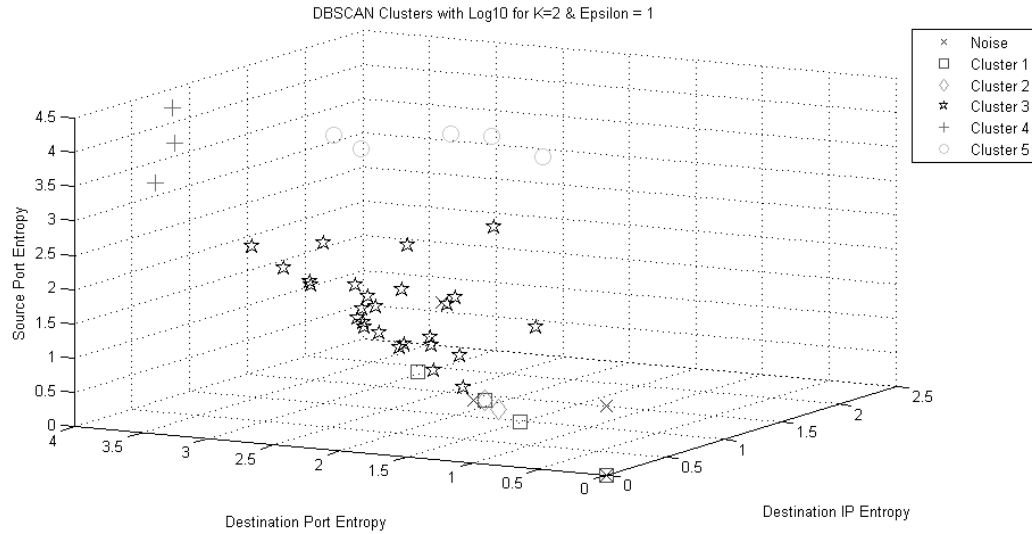
a combination of all the selected features, along with their maximum and minimum values, which are shown for each feature separately.

**Table 5-2 Scan-14 Anomalies detected by Sqalli et al. [19]**

	Dst Port Ent		Src Port Ent		Dst IP Ent		Total Payload Bytes		Total Packets	
	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max
System Compromise	3.15	3.56	3.065	4.465	0	1.84	4.0834	4.1452	2.1398	2.1818
Malicious File Download	X	X	X	X	X	X	4.2254	4.8481	2.1613	2.2671
Running Various Commands	0	2.71	1	3.85	0	1.95	3.1179	3.9364	1.6532	2.0644

**Table 5-3 Scan -14 DBSCAN Clustering with Min-Max Values**

	Dst Port Ent		Src Port Ent		Dst IP Ent		Total Payload Bytes		Total Packets	
	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max
Cluster 1	0	1.42	0	1.25	0	0	0	0	0	0
Cluster 2	0.81	0.92	0.81	0.92	0	0	1.79	1.87	0	0
Cluster 3	1.08	2.66	1.08	3	0	1.48	3.37	4.85	1.11	2.74
Cluster 4	3.25	3.64	3.66	4.49	0	0.44	4.92	5.31	2.66	2.82
Cluster 5	2.43	3.26	3.06	3.67	1.4	2.31	4.61	5.47	2.1	2.79



**Figure 5-3 Scan-14 3-D Clusters**

### 5.2.2 Scan 19

Scan 19 is the smallest trace used by Sqalli et al. [19], where three malicious activities were reported. DBSCAN created four clusters based on the values of the five features used. Figure 5-4 shows the graphical output of the clusters for this trace. Out of these four clusters, Cluster 3 represents a “System Compromise”, while Cluster 4 represents two malicious activities, i.e., “Malicious File Download” and “Port Scan”. The reason for having two events in one cluster is that the minimum and maximum values for this cluster match with two threshold intervals presented by Sqalli et al. [19]. This problem of having a single cluster representing multiple anomalies is due to the fact that Sqalli et al. [19] used separate names for different anomalies falling under the category of a system compromise attack. System compromise attacks always involve exploiting some legitimate service with vulnerability by sending malicious requests to gain access. Most of the attacks which fall under the system compromise category show very similar behavior. The only difference is that as each service has its own vulnerabilities, the

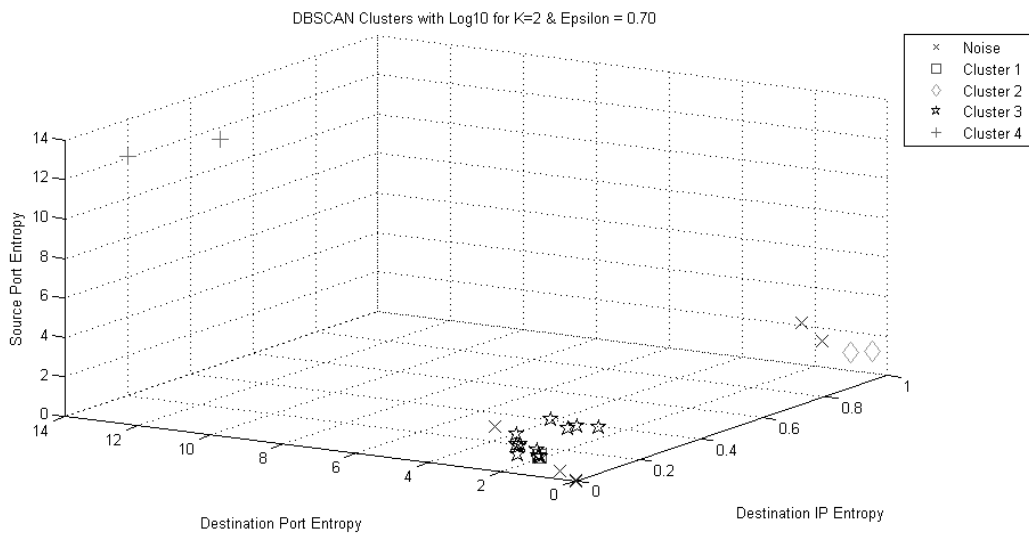
number of malicious messages and size of those malicious messages is different for each attack. These anomalies are almost alike and their threshold values overlap for more than one selected feature. On the other hand, DBSCAN sometimes fails to identify the difference between these anomalies and it clusters them into a single anomaly. This happens due to the similar values for more than one selected feature. Table 5-4 presents the anomalies reported by Sqalli et al. [19] and Table 5-5 presents the DBSCAN Clustering results with the minimum and maximum values.

**Table 5-4 Scan-19 Anomalies detected by Sqalli et al. [19]**

	Dst Port Ent		Src Port Ent		Dst IP Ent		Total Payload Bytes		Total Packets	
	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max
System Compromise	1.8078	1.8078	2.159	0.9893	3.0759	4.11	5	4.1452	1.5185	2.0086
Malicious File Download	X	X	X	X	X	X	5.5729	5.5729	3.6182	3.6185
Port scan	1.5	12.26	1.53	12.26	0.218	0.419	5.3306	5.3306	3.7302	2.0644

**Table 5-5 Scan -19 DBSCAN Clustering with Min-Max Values**

	Dst Port Ent		Src Port Ent		Dst IP Ent		Total Payload Bytes		Total Packets	
	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max
Cluster 1	1	1	1	1	0	0	2.13	2.92	0.3	0.3
Cluster 2	0.37	0.98	0.98	1.2	0.99	0.99	3.71	3.77	1.11	1.11
Cluster 3	0.96	1.91	0.92	1.97	0	0.22	2.51	4.4	0.78	2.44
Cluster 4	12.59	12.97	12.58	12.96	0.08	0.33	5.83	5.86	3.91	4.02



**Figure 5-4 Scan-19 3-D Clusters**

### 5.2.3 Scan 28

Sqalli et al. [19] reported 4 malicious activities in the Day1 of the scan 28, while DBSCAN created five clusters. Figure 5-5 shows the graphical output of the clusters for this trace. Clusters 4 and 5 map exactly to the “Malicious File Download” and “System Compromise” activities, respectively. Cluster 1 is a large cluster representing two malicious activities, i.e., “IRC Communication” and “ICMP (DDoS)”. The reason of having two anomalies in the same cluster is discussed in section 5.2.2. Clusters 2 and 3 show traffic patterns which are not reported as malicious activities by Sqalli et al. [19]. The problem of having a cluster not representing any anomaly is discussed in section 5.2.1. Table 5-6 presents the anomalies reported by Sqalli et al. [19], and Table 5-7 presents the DBSCAN Clustering with the minimum and maximum values.

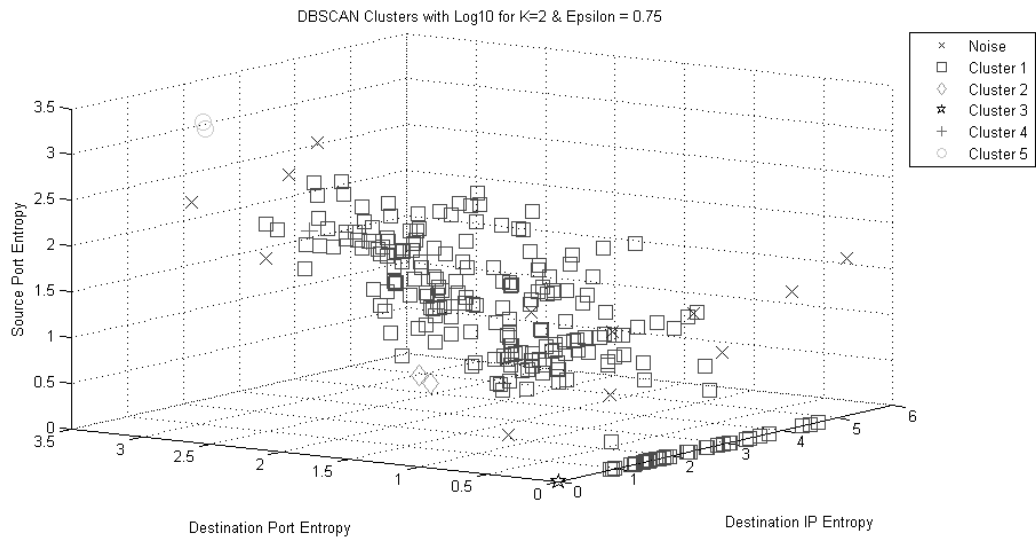
**Table 5-6 Scan-28 Day-1 Anomalies detected by Sqalli et al. [19]**

	Dst Port Ent		Src Port Ent		Dst IP Ent		Total Payload Bytes		Total Packets	
	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max
System Compromise	2.02	2.988	2.02	3.11	0	2.2	3.6577	5.87	1.3424	3.17
Malicious File Download	X	X	X	X	X	X	5.5936	5.87	2.87	3.17
IRC Communication	1	2.5	1	2.6	0	2.5	3.7923	4.279	1	1.9867
ICMP (DDoS)	0	1.38	0	1.63	0.721	3.4	3.8026	4.208	0.7781	1.76
Port Scan	7.09	8.8685	6.95	9.81	0	0.91	5.185	5.883	2.6085	3.5

**Table 5-7 Scan -28 Day-1 DBSCAN Clustering with Min-Max Values**

	Dst Port Ent		Src Port Ent		Dst IP Ent		Total Payload Bytes		Total Packets	
	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max
Cluster 1	0	2.56	0	2.56	0.67	4.97	2.26	4.69	0.3	2.07
Cluster 2	0.92	1	0.92	1	0	0	2.43	2.47	0.3	0.3
Cluster 3	0	0	0	0	0	0	2.4	2.48	0.3	0.6
Cluster 4	2.02	2.19	2.08	2.25	0.99	1.58	5.59	5.81	2.88	3.14
Cluster 5	2.95	3.19	3.19	3.23	1.03	1.57	5.7	5.88	3.03	3.17





**Figure 5-5 Scan-28 Day-1 3-D Clusters**

For the traffic from Day 3 of the scan 28, Sqalli et al. [19] reported 5 malicious activities. When we applied DBSCAN to the output, it created seven clusters. Figure 5-6 shows the graphical output of the clusters for this trace. Cluster 1 represents two malicious activities, i.e., “System Compromise” and “ICMP (DDoS)” due to the overlapping threshold values, explained in section 5.2.2. Clusters 4 and 7 represent “Malicious File Download” and “IRC Communication”, respectively. Clusters 2, 3, and 5 represent a “Port Scan” event. The reason for having three separate clusters is that each one represents a different port scan attack with a different level of intensity. Most of the “Post Scan” attacks consist of very high values of Destination port entropy and Destination IP entropy. When we have multiple “Port Scan” attacks in a single dataset, it is uncommon for all of these attacks to have the same values for port entropy and

Destination entropy. And, if the difference between the entropy values of each attack is more than the  $Eps$ -radius, then they will be classified into separate clusters. Cluster 6 represents an un-reported traffic pattern, due to the same explanation provided in section 5.2.1 .

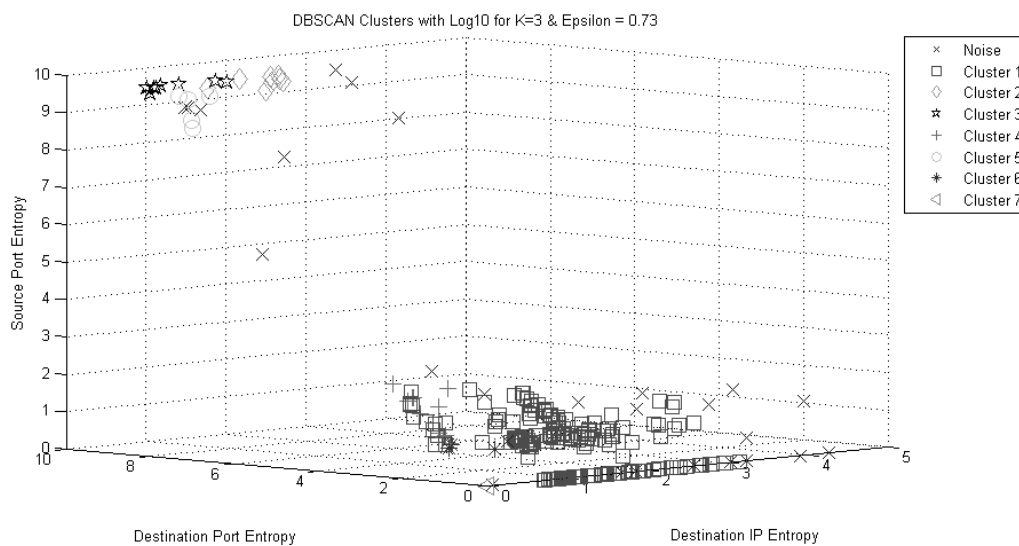
Table 5-8 presents the anomalies reported by Sqalli et al. [19], and Table 5-9 presents the DBSCAN Clustering results with the minimum and maximum values.

**Table 5-8 Scan-28 Day-3 Anomalies detected by Sqalli et al. [19]]**

	Dst Port Ent		Src Port Ent		Dst IP Ent		Total Payload Bytes		Total Packets	
	Min	Max	Min	Max			Min	Max	Min	Max
System Compromise	2.02	2.988	2.02	3.11	0	2.2	3.6577	5.8706	1.3424	3.17
Malicious File Download	X	X	X	X	X	X	5.5936	5.87	2.87	3.17
IRC Communication	0	0	0	0	0	0	3.2193	3.9371	1.176	1.875
ICMP (DDoS)	0	1.38	0	1.63	0.721	3.4	3.8026	4.2088	0.7781	1.7634
Port Scan	4.99	7.424	5.29	9.61	0	0.39	4.748	5.2284	2.8286	3.4429

**Table 5-9 Scan -28 Day-3 DBSCAN Clustering with Min-Max Values**

	Dst Port Ent		Src Port Ent		Dst IP Ent		Total Payload Bytes		Total Packets	
	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max
Cluster 1	0	2.18	0	2.25	0	3.15	3.15	4.66	0.78	2.36
Cluster 2	7.31	8.48	9.61	9.87	0.96	1.57	3.88	4.15	1.18	1.69
Cluster 3	7.96	8.45	9.69	9.87	0.03	1.19	5.19	5.88	3.19	3.49
Cluster 4	1	2.45	1	2.45	0.04	0.68	5.21	5.87	3.14	3.5
Cluster 5	7.04	7.45	8.86	9.68	0.02	0.34	4.07	4.55	1.38	2.12
Cluster 6	0.78	0.98	0.78	0.98	0	0.49	4.57	4.77	2.33	2.87
Cluster 7	0	0	0	0	0	0	3.24	4.2	1.18	2.02



**Figure 5-6 Scan-28 Day-3 3-D Clusters**

### 5.3 RESULTS AND DISCUSSION

Once we have created the clusters, we calculated the maximum and minimum values for each cluster, and then we applied the known anomaly threshold values to these ranges to find the type of anomalous activity represented by that cluster. During these comparisons, we find four types of results.

- The first case is when the cluster boundaries coincide with the anomaly threshold.
- The second case is when a single cluster represents multiple anomalies, and this happens when anomaly thresholds are overlapping with each other in some or all traffic parameters. This happens mostly when the difference between two malicious activities is smaller than the *Eps*-radius, or when their threshold ranges overlap with each other. To overcome this problem, we need to look into the threshold values and come up with a common name for anomalies which have overlapping threshold values or we can add another feature which will help us to better differentiate these types of anomalies.
- The third case is when multiple clusters represent a single anomaly, and this mostly happens due to the presence of multiple attacks of the same type but of different intensity. When we have multiple attacks of the same type in a single dataset, it is not necessary for all of these attacks to have the selected entropy and volume features. And, if the difference between the selected feature values of each attack is more than the *Eps*-radius, then they will be classified into separate clusters.

- The fourth case is when the cluster does not match any of the known threshold values for the anomalies, and this happens mostly due to the presence of non-anomalous traffic in the dataset or the presence of a new type of attacks for which a threshold is not defined. Another reason is the presence of broadcast traffic and in some cases due to the switch level datasets. These datasets do not only contain the packets related to the honeypot traffic but they also have some packets which are not related to the honeypot.

TABLE 5-10 presents the comparison between the reported malicious activities by Sqalli et al. [19] and the clusters which are detected by the DBSCAN Clustering in our work. For Scan 14, DSBSCAN successfully detected all of the anomalies, which were reported by Sqalli et al. [19] as separate clusters. For Scan19, Sqalli et al. [19] reported four anomalies, while DBSCAN was able to detect three anomalies. DBSCAN detected one anomaly as a separate cluster and two anomalies as a part of a larger cluster. For Scan28 day1, Sqalli et al. [19] reported four anomalies, while DBSCAN was able to detect all four anomalies. DBSCAN detected two anomalies as separate clusters and two anomalies as part of larger cluster. For Scan 28 day3, Sqalli et al. [19] reported five anomalies, while DBSCAN was able to detect all five anomalies. DBSCAN detected two anomalies as a separate cluster, two as a part of larger cluster, and one anomaly is represented by three separate clusters.

**Table 5-10 Results Comparison**

Trace	Scan14	Scan19	Scan28 day 1	Scan28 day 3
Reported Malicious Activities in [19]	3	3	4	5
Identified Clusters using DBSCAN	5	4	5	7
Clusters Matching Malicious Activities	3	1+1a	2+1a	2+1a+3b
Clusters not representing Malicious Activities	2	2	2	1

X<sup>a</sup> = One Cluster Represents Multiple Events  
X<sup>b</sup> = Multiple Clusters Represent a Single Event

## CHAPTER 6

### Hierarchical Clustering

Agglomerative Clustering [31] is a Hierarchical Clustering algorithm which we are going to use for this thesis work. Details of Agglomerative Clustering have been discussed in detail in Chapter 4. In this chapter we going to implement Agglomerative Clustering and then we will test the performance of Agglomerative Clustering on different network trace files.

#### 6.1 IMPLEMENTATION

The implementation of the agglomerative clustering algorithm can be divided into three major activities. The first step is to create the distance matrix. The distance matrix for  $N$  data points is an  $N \times N$  matrix consisting of  $N$  rows, where each row represents the distance of one data point to all the other data points. These distances can be calculated using various distance calculation algorithms, e.g., Euclidian distance, city-block distance, etc.

The second step is to create the linkage matrix from the distance matrix. The linkage matrix uses a predefined linkage method, e.g., single-linkage, multiple linkage, average distance, or centroid. The task of the linkage matrix is to calculate the combination

similarity for each data point, and all future clustering operations will be based on this linkage matrix. Based on these combination similarities, a Hierarchical binary tree, called dendrogram is created. A dendrogram can be used to display the Hierarchical binary tree, displaying all the merge operations converging at the root of the binary tree. Figure 4-3 and Figure 4-4 show the dendrogram created from the linkage matrix created for scan 14.

The third and final step is to create clusters from the linkage matrix. For the cluster creation, we can use one of the three cutoff criteria for cluster creation, discussed in section 4.5.1 , i.e., by natural clustering, by specifying a certain threshold value of the combination similarity, or by specifying the number of clusters. To display the output of the clustering process, we can use a 3-D scatter plot.

## **6.2 INITIAL EXPERIMENTS**

For the experimentation purpose, we are using the PCAP files from the “scan of the month” challenges provided by the global HoneyNet community. Once we have completed the implementation, we decided to review different configurations of the agglomerative clustering, and find the combination which will help us to create more informative and well-defined clusters. The most important configuration in agglomerative clustering is the type of linkage used to create the clusters. We will experiment with both average linkage and centroid linkage. We have selected these two linkages because they use all the data points inside the cluster to calculate for the next merge operation. Then, at the end, we will compare the results. For the clustering criteria, we will use natural clustering.



## 6.2.1 Average Linkage Clustering

In the average linkage, when we need to take a decision about the merge operations, we look at all the distance pairs in both clusters and take their average. The average linkage has been discussed in details in section 4.5.2

We used scan-14 and scan-28 for the initial testing with the average linkage clustering.

### 6.2.1.1 Scan-14

The average linkage clustering produced five clusters from the scan-14 dataset, while using the natural clustering criteria. Figure 6-1 shows the 3-D plot of the clusters, and Table 6-1 shows the minimum and maximum values for each cluster. The clusters created by the average linkage clustering do not match with the already known malicious threshold. As discussed in section 4.5.2, the average linkage clustering uses a weighted average to decide about the merge operations. When we applied the average linkage clustering to the scan-14 dataset, it produced five different clusters; but the clusters ranges do not match any of the threshold values for the known anomalies detected by Sqalli et al. [19].

Our understanding about the average linkage clustering's failure to create usable clusters is the use of a weighted average. When using a weighted average, each merge operation tends to favor the larger of the two clusters involved in the merging operation. In the end, clusters created by using average linkage do not match with known threshold values.

Table 6-1 Scan-14 Clusters with min-max values

	Dst Port Ent		Src Port Ent		Dst IP Ent		Total Payload Bytes		Total Packets	
	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max
Cluster 1	0	1.418	0	1.246	0	0	0	0	0	0
Cluster 2	1.076	2.664	1.0762	3.002	0	1.476	3.3672	4.8522	1.1139	2.7443
Cluster 3	0	0	0	0	0	0	4.3375	4.3375	2.4472	2.4472
Cluster 4	0	1.240	0.8113	2.299	0	0	1.7924	2.7536	0	0.6021
Cluster 5	2.426	3.640	3.0586	4.485	0	2.307	4.606	5.4738	2.0969	2.8176

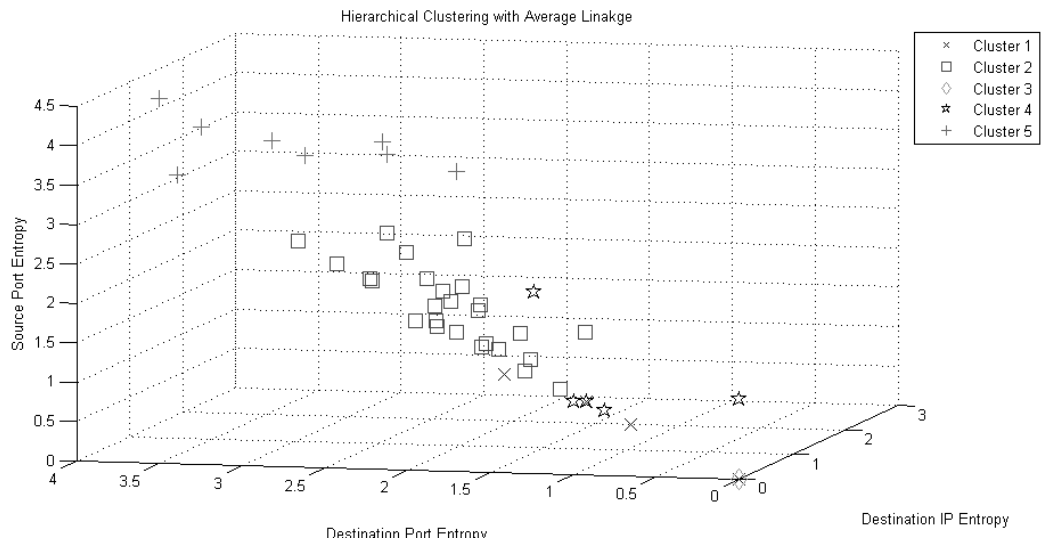


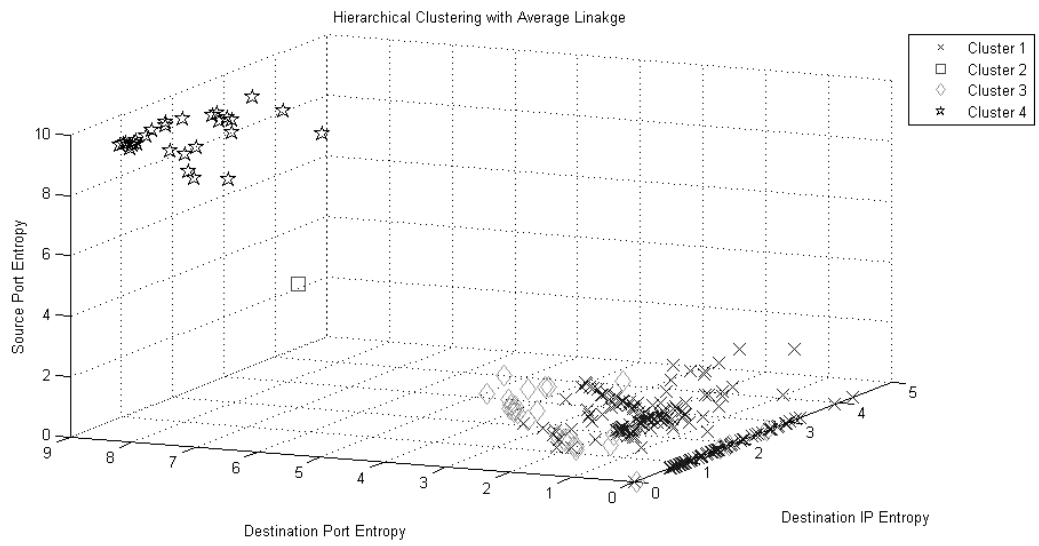
Figure 6-1 Scan-14 Hierarchical clustering with average linkage

### 6.2.1.2 Scan-28 Day-3

The average linkage clustering produced four clusters when applied to the scan-28 day-3 dataset. The natural clustering is used here as the clustering criteria. Table 6-2 shows the clusters with their minimum and maximum ranges. Figure 6-2 shows the 3-D scatter plot for the scan-28 day-3. Similar to the previous trace, the clusters created by the average linkage clustering for this trace do not match with the already known malicious activities ranges. The reason for not matching any known threshold value is discussed in section 6.2.1.1.

**Table 6-2 Scan 28 Day-3 Clusters with min-max values**

	Dst Port Ent		Src Port Ent		Dst IP Ent		Total Payload Bytes		Total Packets	
	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max
Cluster 1	1.53	1.98	1.53	1.98	1.08	1.25	5.8	6.17	3.01	3.36
Cluster 2	0	2.39	0	2.26	0	3.34	3.46	4.99	1.04	2.64
Cluster 3	7.02	9.52	7.87	10.19	0.02	0.44	4.97	6.18	2.91	3.81
Cluster 4	1.04	2.68	1.04	2.68	2.58	4.58	4.29	4.97	1.85	2.72



**Figure 6-2 Scan-28 Day-3 Hierarchical clustering with average linkage**

## **6.2.2 Centroid Linkage Clustering**

In the centroid linkage, when we need to take a decision about the merge operation, we first calculate the centroid or center of mass, which is the average of all the data points inside the cluster. Then we calculate the minimum distance between centroids. All the decisions about the merge operations are based on the minimum distance between the centroids. Centroid Linkage Clustering has been discussed in details in section 4.5.2

We used scan-14 and scan-28 for the initial testing with the centroid linkage clustering.

### **6.2.2.1 Scan-14**

The centroid linkage clustering produced four clusters when applied to the scan-14 dataset, while using the natural clustering as the clustering criterion. Table 6-3 shows the minimum and maximum values of the features in a clusters and it shows the type of anomaly which these clusters are representing. Figure 6-3 shows the 3-D scatter plot for scan-14.

Table 6-3 Scan-14 Clusters with min-max values

	Dst Port Ent		Src Port Ent		Dst IP Ent		Total Payload Bytes		Total Packets	
	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max
Cluster 1-RVC	1.08	2.66	1.08	3	0	1.48	3.37	4.85	1.11	2.74
Cluster 2-MFD	0	0	0	0	0	0	4.34	4.34	2.45	2.45
Cluster 3-noise	0	1.42	0	2.3	0	0	0	2.75	0	0.6
Cluster 4-SC	2.43	3.64	3.06	4.49	0	2.31	4.61	5.47	2.1	2.82

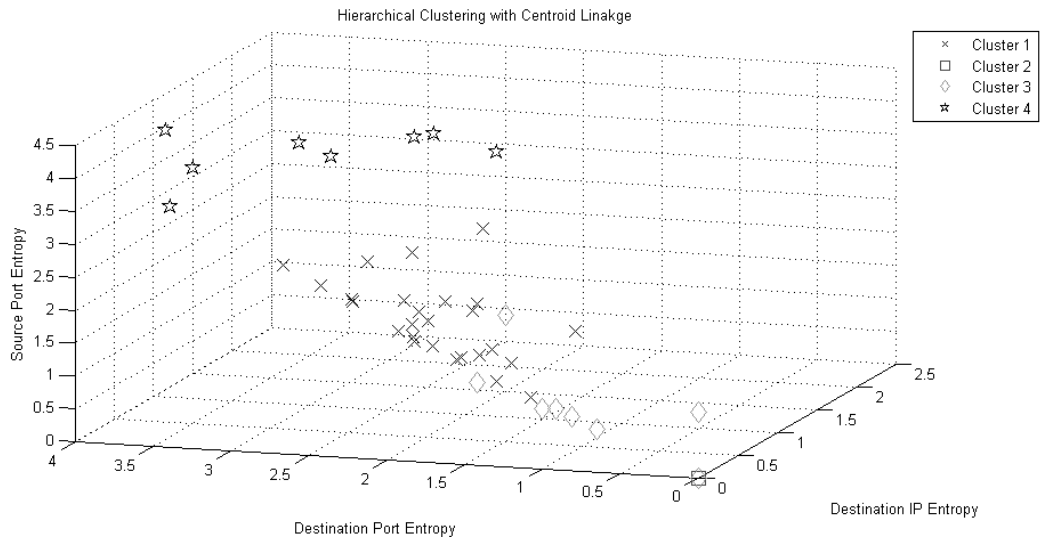


Figure 6-3 Scan-14 Centroid Linkage Clustering

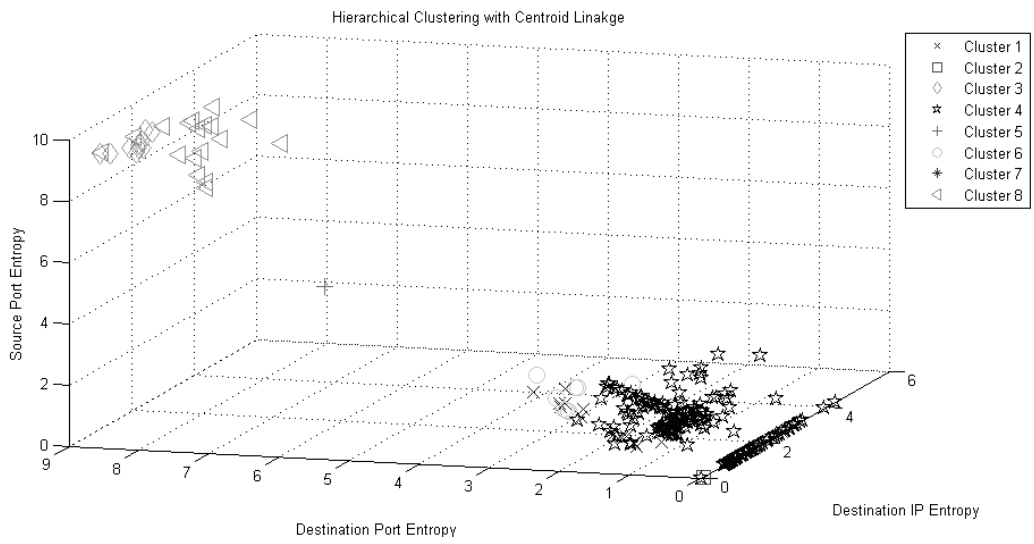
### 6.2.2.2 Scan-28 Day-3

The centroid linkage clustering produced eight clusters when applied to the scan-28 day-3 dataset, while using the natural clustering as the clustering criterion.

Table 6-4 shows the minimum and maximum values of the clusters as well as the type of anomaly represented by these clusters. Figure 6-4 shows the 3-D scatter plot for scan-28 Day-3.

**Table 6-4 Scan-28 Day-3 Clusters with min-max values**

	Dst Port Ent		Src Port Ent		Dst IP Ent		Total Payload Bytes		Total Packets	
	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max
Cluster 1-IRC	0	0	0	0	2.22	3.03	4.09	4.49	1.32	1.87
Cluster 2-s-PS	9.42	9.52	9.54	9.58	0.06	0.08	6.17	6.18	3.47	3.49
Cluster 3-s-PS	7.02	7.98	7.87	8.93	0.34	0.44	4.97	5.71	2.91	3.13
Cluster 4-SC	0.41	2.68	0.65	2.68	0.49	3.82	3.46	4.99	1.23	2.72
Cluster 5-s-MFD	1.53	1.98	1.53	1.98	1.08	1.25	5.8	6.17	3.01	3.36
Cluster 6-s-SC	1.04	1.68	1.04	1.68	3.9	4.58	4.33	4.42	1.9	2.18
Cluster 7-ICMP	0	0.92	0	0.57	0	1.99	3.46	4.54	1.04	2.44
Cluster 8-PS	7.26	8.82	9.42	10.19	0.02	0.18	5.2	5.99	3.34	3.81



**Figure 6-4 Scan-28 Day-3 Centroid Linkage Clustering**



## 6.3 RESULTS DISCUSSION

In this chapter, we experimented with a hierarchical Clustering algorithm with two different linkages, average linkage and centroid linkage. In the average linkage, we were able to create different clusters but these clusters did not match any of the detected threshold values for the known anomalies. On the other hand, the centroid linkage clustering was able to identify all of the anomalies, available in the dataset, by creating the clusters that match the detected threshold values for the known anomalies. Based on the results obtained here, we decided to use the centroid linkage for the experiments related to the hierarchical clustering in this thesis work. In summary, the hierarchical clustering with centroid linkage worked very well in identifying the general attack categories. But in some cases, especially for the system compromise attack, it failed to do so; while Sqalli et al. [19] have divided the thresholds into specific attack categories. For these attacks, which fall under the sub-category of system compromise attack, Hierarchical clustering sometimes categorized the anomalies into their generalized attack category, i.e., system compromise, rather than creating separate clusters for each specialized attack.

## **CHAPTER 7**

### **Performance Analysis**

In this chapter, we will apply both clustering algorithms, DBSCAN and Hierarchical Clustering, to different datasets. Some of these datasets are provided by the HoneyNet community and some are generated inside a KFUPM lab. Once we have the results, by applying both schemes to the same datasets, then we will compare the efficiency of both algorithms as well as the detection rate. This will also help to determine which algorithm is better to be used for the automatic detection of the anomalies from new datasets.

#### **7.1 EXPERIMENTAL SETUP**

For the experimental setup, we are using a JAVA code to determine the entropy values for the selected features from the provided PCAP files. JNetPcap Java API is used to read the PCAP files. Once we have the entropy values, then we will provide these entropy values as an input to both algorithms to create the clusters. Both algorithms will provide the output in the form of 3-D scatter plots and cluster tables, comprising the minimum and maximum values for each cluster against each of the selected features.

The trace files used for obtaining the results are:

- Scan 27: Scan of the month challenge provided by honeynet.org, March 2003
- Dionaea Capture Trace-1 and 2: This is provided by the Saudi Honeynet Project team, collected from the KFUPM network.
- Lab Trace: Trace created inside a lab environment by using the penetration testing tool provided in backtrack 4.1 [21].

## 7.2 DESCRIPTION OF TRACES USED

### 7.2.1 Scan 27

This trace was collected from Honeynet.org which releases the Scan of the Month Challenges. This trace was collected by the Azusa Pacific University Honeynet Project team from an un-patched Windows 2000 honeypot. The details of this trace are provided in Table 7-1.

**Table 7-1 Scan-27 Dataset Details**

Source	Honeynet.org, Scan of the Month Challenge
File Name	Scan27.pcap
Format	PCAP File
Size	17.6 MB
Number of Packets	54536
Duration	5 Days

The identification of anomalies inside this dataset was also provided by the HoneyNet.org as the solution for this challenge. We will compare our results against the solutions provided by the HoneyNet.org.

### 7.2.2 Dionaea Capture Trace

A low interaction honeypot Dionaea was setup and connected in the KFUPM network. The two trace files were collected by the Saudi HoneyNet Project team as part of their network monitoring project. The trace details are given in Table 7-2.

**Table 7-2 Dionaea Dataset Details**

Source	Saudi HoneyNetProject team
Number of files	2
File Name	Dionaea-trace1.pcap
File Name	Dioanea-trace2.pcap
Format	PCAP File
Size	15.6 MB, 648KB
Number of Packets	1541731
Duration	1 Day

The result of this dataset is provided by the Saudi HoneyNet Project team. We will use these results to compare them to the clusters created by both algorithms used in our work.

### 7.2.3 Lab Trace

The last trace that we have used was generated in the Lab setup within KFUPM. A Honeynet was setup with Honeywall - a high interaction honeypot and Windows XP honeypot. The BackTrack 4.1 operating system was used to launch different types of attacks targeting the Windows XP honeypot. The honeypot was made visible on the network and popular services were activated on it such as IIS web server, FTP server, SSH server, etc. The main tools that were used from the BackTrack operating system are Nmap, Open VAS vulnerability scanner, and Metasploit Penetration Testing Framework 3.0.

The Metasploit Framework [30] is one of the most popular Open Source penetration testing tools that are available in the market [34]. We used these tools to generate a trace that includes different types of malicious activities. The Metasploit framework has been used by other authors to generate similar datasets for the purpose of evaluating their anomaly detection techniques. Laskov and Kloft [35] have used the Metasploit framework to create a malicious dataset by generating various exploits. Rieck and Laskov [36] have also used the Metasploit framework to create a malicious dataset for the purpose of testing their anomaly detection technique. Düssel et al. [37] also used the Metasploit framework to generate malicious dataset for testing their anomaly detection technique.

Details of the trace file used are given in the Table 7-3.

**Table 7-3 Lab Trace Dataset Details**

Source	Network Security Lab
File Name	Labcptr.pcap
Format	PCAP File
Size	30 MB
Number of Packets	312599
Duration	22 Day

The details of the attacks conducted are presented in Table 7-4.

**Table 7-4 Lab Trace Attacks Detail**

<b>Categories</b>	<b>Type of attack</b>
Port Scan	NMAP regular scan NMAP quick scan NMAP intense scan NMAP slow comprehensive scan
Vulnerability Scanning	Open VAS Scanner
Database attacks	MYSQL login utility scanner MYSQL database access attempts
Server Message Block (SMB) protocol attacks	SMB Negotiate Dialect Corruption (Fuzzers/smb/smb_negotiate_corrupt) Microsoft Workstation Service NetAddAlternateComputerName Overflow Microsoft Server Service Relative Path Stack Corruption Microsoft Server Service NetpwPathCanonicalize Overflow Microsoft Plug and Play Service Overflow Microsoft Print Spooler Service Impersonation Vulnerability
DCE/RPC, (Distributed Computing Environment / Remote Procedure Calls) attacks	Endpoint Mapper Service Discovery (scanner/dcerpc/endpoint_mapper) DCERPC TCP Service Auditor Microsoft RPC DCOM Interface Overflow exploit Microsoft Message Queuing Service Path Overflow exploit

FTP	Simple FTP Fuzzer FTP attack access gain attempt
HTTP IIS web server attacks	Microsoft IIS WebDAV Writ exploit Microsoft IIS 5.0 Printer exploit Microsoft IIS/PWS CGI Fil exploit Microsoft IIS 5.0 WebDAV ntdll.dll Path Overflow
SMTP attacks	MS03-046 Exchange 2000 XEXCH50 Heap Overflow exploit
SNMP attacks	Network Node Manager Snmp.exe CGI Buffer Overflow
Backdoor	Energizer DUO Trojan Code Execution
SSH attacks	SSH Key Exchange Init Corruption

### 7.3 COLLECTED RESULTS

We applied the two clustering algorithms, Hierarchical clustering and DBSCAN clustering, to the datasets discussed in the previous section. The process of gathering results is similar to the process discussed in chapters 4 and 5 for DBSCAN Clustering and Hierarchical Clustering, respectively. Initially, we calculated the entropy value for the three IP header based traffic features, i.e., Destination port entropy, Source port entropy, and Destination IP entropy. And, for the two volume based features, i.e., total payload bytes and total packet count, we took the  $\log_{10}$  of the original values for normalization. The values of the entropy based features were in the range of one to fifteen, while the values of the volume based features were in the ranges of five hundred to ten thousand. Therefore, we have normalized the volume values by taking  $\log_{10}$  of the



original values. Once we have all the values of these five traffic features stored in a comma separated file, we use them as an input to the clustering algorithms.

The results collected from each clustering algorithm are provided in this chapter. The results will be presented in the following order. First, we will present the results for each dataset separately; and then at the end, we will discuss all the collected results and compare the performance of both clustering algorithms, i.e., DBSCAN and Hierarchical Clustering. For each dataset, initially we will provide all the reported results from the literature, and then we will present our results for that dataset using both Hierarchical and DBSCAN Clustering. Finally, we will compare our collected results with the reported results for that dataset.

### **7.3.1 Scan 27**

Scan of the month challenge 27 was presented by the honeynet.org as a challenge of the month. Table 7-5 shows the reported anomalies for the scan 27.

**Table 7-5 Reported Anomalies for Scan-27**

<b>Type of Anomaly</b>	<b>No. of Occurrences</b>
SMB Attacks	5
System Compromise	1
Malicious File Download	1
HTML Script Kiddies	1
Buffer Overflow	1
Port Scan	3
Vulnerability Scan	1
Slammer Worm	1
IRC Communication	1

### 7.3.1.1 Hierarchical Clustering

Hierarchical clustering produced nine clusters when applied to the scan 27 datasets. Out of these nine clusters, clusters 1, 4, and 6 are all port scan attacks. Cluster 9 matches the threshold of the vulnerability scan. Cluster 7 matches the threshold of the IRC communication from the affected machine to the botnet server. Clusters 2 and 8 match the threshold values for the system compromise attack. Clusters 3 and 5 match the threshold values for the system compromise attack. Cluster 3 matches the malicious file download's threshold signatures. On the other hand, cluster 5 presents the noise inside the input dataset, which is mostly due to the fact that these scan files also have some network traffic which is not coming from or going to the Honeypot IP address. Scan 27 is a switch level trace file, so it also includes traffic coming to other hosts connected to the same switch. Figure 7-1 shows the 3-D plot for the clusters created by the Hierarchical clustering algorithm. Table 7-6 shows the cluster ranges for each type of anomaly.

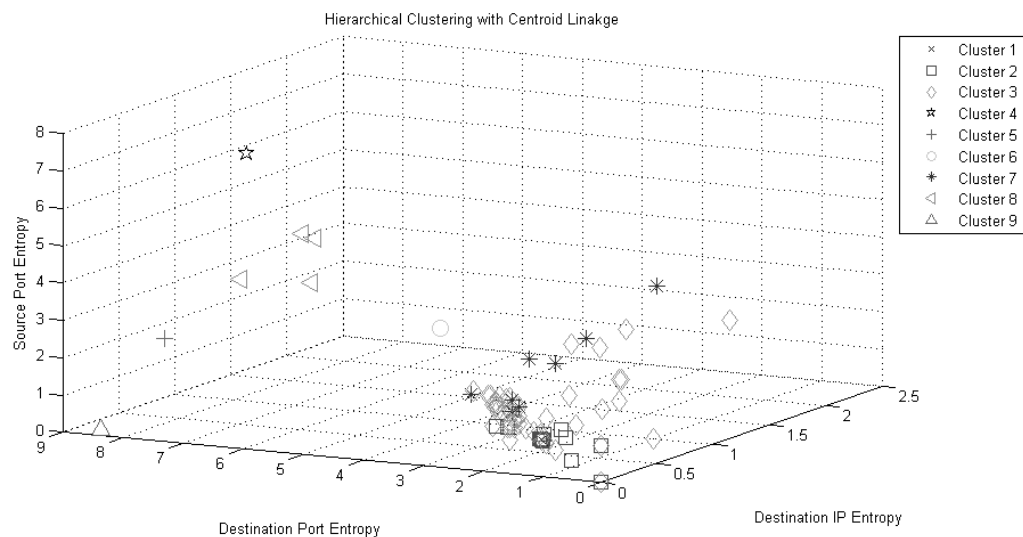


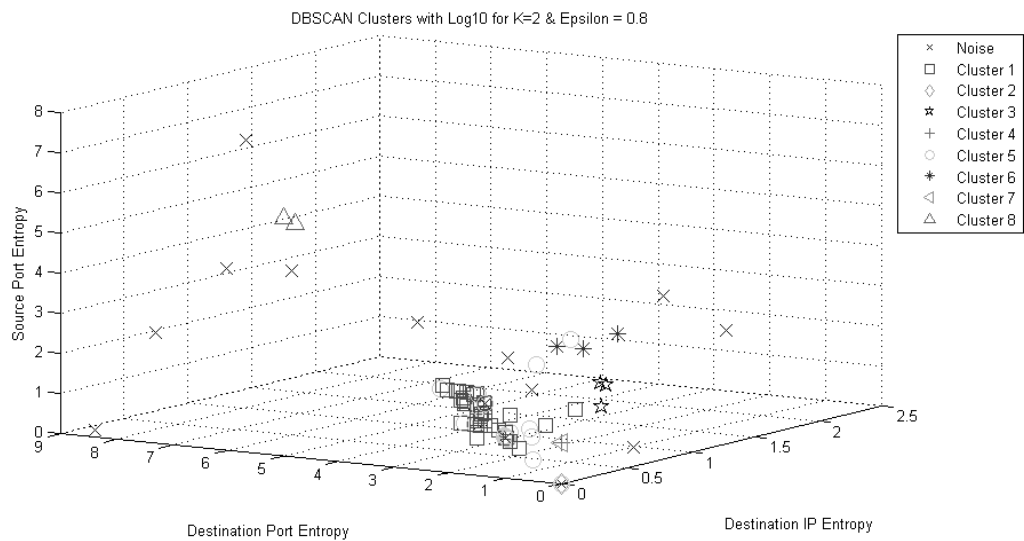
Figure 7-1 3-D Hierarchical Clustering of scan 27

**Table 7-6 Hierarchical Clustering of scan 27 with Min-Max Values**

	Dst Port Ent		Src Port Ent		Dst IP Ent		Total Payload Bytes		Total Packets	
	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max
Cluster 1-PS	7.33	7.33	2.74	2.74	0.02	0.02	5.25	5.25	3.44	3.44
Cluster 2-SC	0	2.38	0.88	2.48	0	1.9	2.18	3.82	0.3	1.58
Cluster 3-MFD	1.64	1.64	1.41	1.41	1.06	1.06	7.06	7.06	4.23	4.23
Cluster 4-PS	7.09	7.09	7.11	7.11	0.64	0.64	4.33	4.33	2.49	2.49
Cluster 5-noise	0	1.21	0	1.21	0	0.21	0	0	0	0
Cluster 6-PS	8.37	8.37	0.17	0.17	0	0	5.08	5.08	3.3	3.3
Cluster 7-IRC	0	2.27	0	2.42	0	0.55	4.2	6.2	2.18	3.4
Cluster 8-SC	3.04	3.12	2.61	3.41	0.23	2.12	5.51	5.71	2.83	3.07
Cluster 9-VS	4.78	6.02	4.54	5.93	0	0	4.95	5.39	3.1	3.58

### 7.3.1.2 DBSCAN

When we applied DBSCAN clustering to the Scan 27, it produced eight clusters. Clusters 2 and 4 match the threshold of the IRC Communication from a compromised machine to the botnet server. Clusters 5 and 6 match the threshold for the System compromise attack. Cluster 7 matches the malicious file down attack's threshold. Cluster 1 matches the HTML script kiddies attack and cluster 3 matches the slammer worm attack. Cluster 8 matches the threshold of the vulnerability scan attack. Figure 7-2 shows the 3-D plot of the DBSCAN clustering for the Scan 27.



**Figure 7-2 3-D DBSCAN clustering of scan 27**

Table 7-7 represents the clusters created by the DBSCAN clustering along with the minimum and maximum values for each cluster.

**Table 7-7 DBSCAN Clustering of Scan 27 with Min-Max values**

	Dst Port Ent		Src Port Ent		Dst IP Ent		Total Payload Bytes		Total Packets	
	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max
Cluster 1-Html Script	0.76	2.15	0.76	2.15	0	0.54	2.33	3.82	0.3	1.58
Cluster 2-IRC	0	0	0	0	0	0	2.18	3.08	0.3	1.11
Cluster 3-Slammer	1.53	1.64	0.92	1.53	0.97	1.06	3.02	3.14	0.6	1
Cluster 4-IRC	0	0	1	1	0	0	2.92	3.26	0.3	1.04
Cluster 5-SC	0	2.17	0	2.52	0	0.99	4.2	5.44	2.18	3.58
Cluster 6-SC	1.91	2.37	2.31	2.36	1	1.38	2.68	2.92	0.3	0.48
Cluster 7-MFD	0	0	1	1	0	0	5.08	5.51	2.83	3.3
Cluster 8-VS	4.78	4.98	5.82	5.93	0	0	3.02	3.14	1	1

Table 7-8 presents the results comparison of both clustering algorithms against the reported results as well as the detection results of sqalli et al. [19]. The Hierarchical Clustering algorithm was successful in detecting 82% of the anomalies except for the Buffer Overflow attack, the Slammer worm attack, and the HTML script kiddies attack. Hierarchical clustering also performed very well in detecting the port scan attack, according to their intensity and categorized them in separate clusters. DBSCAN does not perform very well on this dataset as it only detected 71% of the anomalies. DBSCAN was unable to detect any of the port scan attacks or buffer overflow attacks. But, it was

successful in detecting the Slammer worm attack and the HTML script kiddies' attack, which had gone undetected in Hierarchical clustering. Both Clustering algorithms are able to detect the system compromise attack and also the SMB attack on Microsoft services running on port 445. Both Clustering algorithms were also successful in detecting malicious file download and IRC Communication attacks.

**Table 7-8 Result Comparison for scan 27**

<b>Type of Anomaly</b>	<b>No. of Occurrences</b>	<b>Hierarchical Clustering</b>	<b>DBSCAN</b>	<b>Sqalli et al[19]</b>
SMB Attacks	5	Yes, one cluster	Yes, one cluster	Yes
System Compromise	1	Yes, one cluster	Yes, one cluster	Yes
Slammer Worm	1	Merged in other Cluster	Yes, one cluster	Not detected
HTML Script Kiddies	1	Merged in other Cluster	Yes, one cluster	Yes
Buffer Overflow	1	Merged in other Cluster	Merged in other Cluster	Yes
Malicious File Download	1	Yes, one cluster	Yes, one cluster	Yes
Port Scan	3	Yes, Three cluster	Not Detected	Yes, 1 occurrence
Vulnerability Scan	1	Yes, one cluster	Yes, one cluster	Not detected
IRC Communication	2	Yes, one cluster	Yes, two cluster	Yes

### 7.3.2 Dionaea Capture Trace-1

The Dionaea Capture Trace-1 was captured by the Saudi HoneyNet Project team as part of their HoneyPot deployment inside KFUPM network. The anomalies for this trace that have been reported by the Saudi HoneyNet Project team in Sqalli et al. [19] are presented in Table 7-9.

**Table 7-9 Reported Anomalies in Dionaea Capture Trace-1**

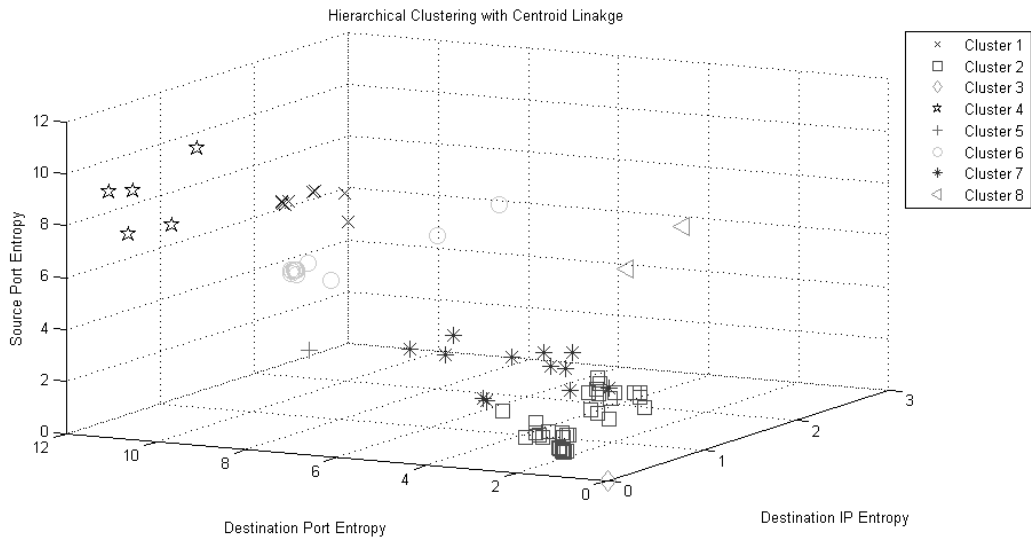
<b>Type of Anomaly</b>	<b>No. of Occurrences</b>
Web Robots	2
System Compromise, SQL Brute Force	2
Connection attempts on popular ports	4
SIP Worm	1
Port Scan	3

#### 7.3.2.1 Hierarchical Clustering

When we applied Hierarchical clustering to the Dionaea Capture Trace-1, we obtained eight clusters as output. Cluster 4 represents the port scan attack. Clusters 5 and 6 represent the system compromise attack on the SQL by brute forcing or password guessing. Clusters 7 and 8 match the threshold of the attack on the popular ports by having multiple connection attempts. Cluster 1 matches the threshold of the web robots attack on the honeypot and cluster 2 represents the SIP worm attack, which tries to connect of the SIP port 5060. Cluster 3 represents the noise in the available dataset. This noise is mostly due to the fact that all the packets available in the dataset are not coming



from or going towards the Honeygot. Figure 7-3 shows the 3-D plot of the Hierarchical clustering on the Dioanea Capture Trace-1.



**Figure 7-3 3-D plot of the Hierarchical Clustering of Dioanea Capture Trace-1**

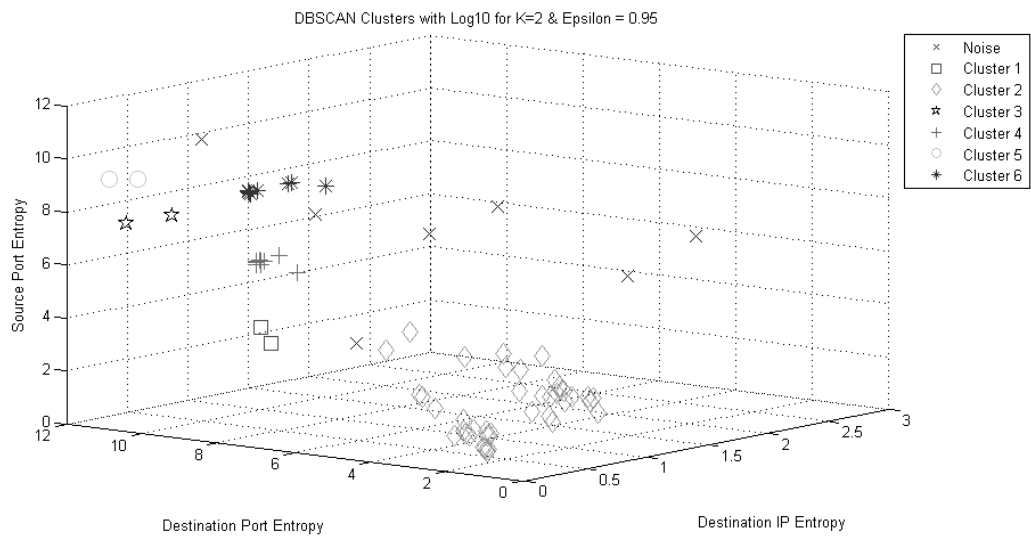
Table 7-10 represents the cluster created by the Hierarchical clustering along with the minimum and maximum values for each cluster.

**Table 7-10 Hierarchical Clustering of Dioanaea Capture trace-1 with Min-Max Values**

	Dst Port Ent		Src Port Ent		Dst IP Ent		Total Payload Bytes		Total Packets	
	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max
Cluster 1-Web Robots	6.37	7.25	8.68	9.66	0	0.57	5.72	6	3.74	4.02
Cluster 2-SIP	0.86	2.32	0.92	2.59	0	1	2.09	4.12	0.3	1.79
Cluster 3-noise	0	0	0	0	0	0	0	0	0	0
Cluster 4-PS	10.3 3	11.4 4	7.56	10.5 4	0.17	0.59	5.3	6.23	3.49	4.42
Cluster 5-SC SQL BF	6.65	6.65	3.98	3.98	0.01	0.01	5.71	5.71	3.82	3.82
Cluster 6-SC- SQL BF	5.64	7.07	6.52	7.96	0	1.56	5.18	5.66	3.18	3.6
Cluster 7-Conection attempts	1.88	4.39	2.2	4.48	0	1.14	2.99	4.21	1.23	2.36
Cluster 8-Connection Attempts	3.65	3.92	5.16	6.28	2.09	2.56	4.67	5.35	2.62	3.15

### 7.3.2.2 DBSCAN

When we applied DBSCAN clustering to the Dionaea Capture Trace-1, it produced six clusters. Clusters 1 and 4 represent the System Compromise attack on the MS-SQL service by brute forcing or password guessing. Cluster 2 matches the threshold of the attack on the popular services by having various connection attempts. Clusters 3 and 5 represent the port scan attack on the Honeypot IP address, while cluster 6 matches the threshold of the web robots attack on the Honeypot. Web robots are mostly automated scripts which crawl over the Internet, looking for vulnerable web servers. Figure 7-4 represents the 3-D plot of the DBSCAN clustering for the Dionaea Capture Trace-1.



**Figure 7-4 3-D plot of DBSCAN clustering of Dionaea Capture Trace-1**

Table 7-11 shows the DBSCAN Clustering of the Dionaea Capture Trace-1 with the minimum and maximum values of each cluster.

**Table 7-11 DBSCAN Clustering of Dionaea Capture Trace -1 with Min-Max Values**

	Dst Port Ent		Src Port Ent		Dst IP Ent		Total Payload Bytes		Total Packets	
	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max
Cluster 1-SC-SQL BF	6.65	6.93	3.98	4.55	0.01	0.01	5.71	5.93	3.82	4.04
Cluster 2- Connection attempts	0.86	4.26	0.92	4.48	0	1.14	2.09	4.21	0.3	2.36
Cluster 3-PS	10.5	11.0	7.56	7.76	0.21	0.42	5.38	5.47	3.6	3.67
Cluster 4--SC-SQL BF	6.52	7.07	6.52	7.07	0	0.2	5.18	5.3	3.18	3.3
Cluster 5-PS	11.2	11.4	9.05	9.21	0.17	0.35	5.84	6.23	4.05	4.42
Cluster 6- Web robots	7.01	7.25	9.36	9.66	0	0.57	5.96	6	3.97	4.02

Table 7-12 represents the results comparison of both clustering algorithms against the reported results as well as the detection results of sqalli et al. [19]. The Hierarchical Clustering algorithm was successful in detecting all of the anomalies. The Hierarchical clustering performed very well in detecting the port scan attack, and SQL brute force attacks. Hierarchical clustering was also able to detect the SIP worm attack. DBSCAN also perform very well on this dataset as it was able to detect 92% of the attacks. It was also successful in detecting port scan attacks and placing them in a separate cluster according to their intensity. However, DBSCAN was unable to detect the SIP worm

attack. Both Clustering algorithms are able to detect the system compromise attack, which was an SQL brute force in this case. Both Clustering algorithms were also successful in detecting the Web robots attack and the multiple connection request attack.

**Table 7-12 Comparison of the reported results with Clustering result**

<b>Type of Anomaly</b>	<b>No. of Occurrences</b>	<b>Hierarchical Clustering</b>	<b>DBSCAN</b>	<b>Sqalli et al, [19]</b>
Web Robots	2	Yes, 1 Cluster	Yes, 1 Cluster	Yes
System Compromise, SQL Brute Force	2	Yes, 2 Clusters	Yes, 2 Clusters	Yes
Connection attempts on popular ports	4	Yes, 2 Clusters	Yes, 1 Cluster	Yes
SIP Worm	1	Yes, 1 Cluster	Merged in other Cluster	Not detected
Port Scan	3	Yes, 1 Cluster	Yes, 2 Cluster	Yes

### **7.3.3 Dionaea Capture Trace-2**

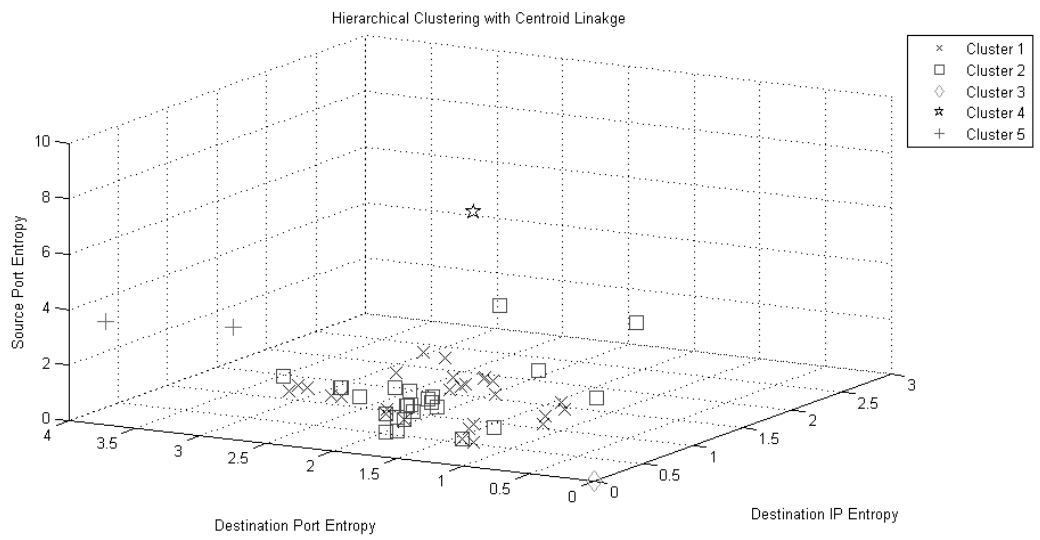
Dionaea Capture Trace-2 was captured by the Saudi HoneyNet Project team as part of their HoneyPot deployment inside the KFUPM network. The following anomalies have been reported for this trace by the Saudi HoneyNet Project team in Sqalli et al. [19], and are presented in Table 7-13.

**Table 7-13 Reported results for the Dionaea Capture Trace-2**

<b>Type of Anomaly</b>	<b>No. of Occurrences</b>
Vulnerability Scanning	1
phpmyadmin Attacks	1
Connection attempts on popular ports	1
SIP Worm	1

### **7.3.3.1 Hierarchical Clustering**

When we applied Hierarchical clustering to the Dionaea Capture Trace-2 trace, we get five clusters as output. Cluster 4 matches the threshold values for the vulnerability scan. Cluster 2 matches the description of the attack targeted towards PhpMyAdmin. Cluster 5 matches the threshold of the attack on the popular port by having multiple connections; and in this dataset, the popular port used was the FTP port. Cluster 1 matches the description of the SIP worm, and cluster 3 is labeled as noise. Noise in the datasets is due to the broadcast traffic and also to the traffic which is not intended for the Honeypot IP address. Figure 7-5 shows the 3-D plot of the Hierarchical Clustering of the Dionaea Capture Trace-2.



**Figure 7-5 3-D plot of the Hierarchical Clustering of Dionaea Capture Trace-2**

Table 7-14 shows the Hierarchical Clustering of the Dionaea Capture Trace-2 with the minimum and maximum values of each cluster.

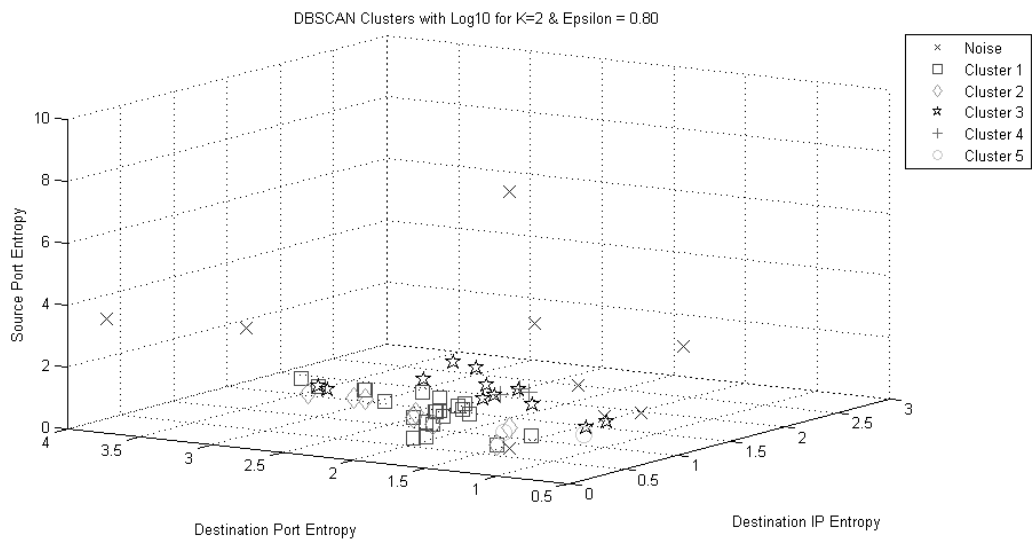
**Table 7-14 Hierarchical Clustering of Dionaea Capture trace-2 with Min-Max Values**

	Dst Port Ent		Src Port Ent		Dst IP Ent		Total Payload Bytes		Total Packets	
	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max
Cluster 1-SIP	0.81	2.32	0.92	2.25	0	1.09	2.06	3.91	0.3	2.06
Cluster 2-PHP MA	0.92	2.41	0.92	2.41	0	2.53	4.39	5.93	2.32	4.12
Cluster 3-noise	0	0	0	0	0	0	0	0	0	0
Cluster 4-VS	0.92	0.92	9.22	9.22	0	0	2.09	2.09	0.3	0.3
Cluster 5-FTP con	3.03	3.72	3.4	3.72	0	0.38	2.26	3.69	0.48	1.85

### 7.3.3.2 DBSCAN

When we applied DBSCAN Clustering to the Dionaea Capture Trace-2, it produced five clusters. Clusters 1, 2, and 3 match the threshold of the attack on the PhpMyAdmin. Cluster 4 represents the SIP attack worm and Cluster 5 represents the attack on the FTP port. Figure 7-6 shows the 3-D clustering created by the DBSCAN from the Dionaea Capture Trace-2.





**Figure 7-6 3-D plot of DBSCAN clustering of Dionaea Capture Trace-2**

Table 7-15 shows the DBSCAN Clustering of the Dionaea Capture Trace-2 with the minimum and maximum values of each cluster.

**Table 7-15 DBSCAN Clustering of Dionaea Capture Trace -2 with Min-Max Values**

	Dst Port Ent		Src Port Ent		Dst IP Ent		Total Payload Bytes		Total Packets	
	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max
Cluster 1-PHP MA	1	2.41	0.92	2.41	0	0.48	4.88	5.93	2.88	4.12
Cluster 2- PHP MA	0.92	2.32	1	2	0	0	2.08	3.28	0.3	1.34
Cluster 3- PHP MA	0.92	2.25	0.92	2.25	0	1.09	3.12	3.85	1.08	1.65
Cluster 4-SIP	1.52	1.59	1.52	1.59	0.81	1	2.06	2.41	0.3	0.6
Cluster 5-FTP Con	0.81	0.95	0.92	1.41	0	0.56	3.85	3.91	1.89	2.06

Table 7-16 represents the comparison of the reported results with the results obtained by both clustering algorithms, as well as the comparison with the results reported by Sqalli et al. [19]. Hierarchical clustering was able to identify all the anomalies present in the Dionaea Capture Trace-2 dataset, including Vulnerability scanning, PhpMyAdmin attack, FTP attack, and the SIP worm attack. Similar to the Hierarchical clustering, DBSCAN Clustering was able to identify all the anomalies in the dataset. DBSCAN reported PhpMyAdmin attacks under 3 different clusters, while in the reported results it has only one occurrence. The reason of having multiple clusters in this case is that the dataset we used for this experiment has very few packets, and all the data points are very sparse. Also, the single occurrence of a PhpMyAdmin attack has similar IP and Port Entropy values, but the values of the total payload bytes and packet count vary during the

whole attack. This is due to the fact that a PhpMyAdmin attack starts with a very small number of packets, in which the attacker uses a specially crafted URL to gain access to the system. After that, the attacker tries to download a rootkit to the compromised PhpMyAdmin server, and this involves a high number of packets and payload bytes. Finally, the attacker uses the rootkit to run various commands on the compromised PhpMyAdmin server, and this involves a medium number of packets and payload bytes. So, all these three clusters lie in the same threshold area, but with little differences. Because of this, DBSCAN has labeled all the three clusters as a single anomaly category, i.e., the PhpMyAdmin attack. DBSCAN was also unable to identify the Vulnerability scan attack on the Honeypot IP. The 3-D graph of the original dataset shows the existence of the data points in the threshold region for the vulnerability scan, but DBSCAN was unable to identify the anomaly because the number of points is less than the number required to create a DBSCAN cluster within the neighborhood radius. For this reason, DBSCAN was not able to identify the Vulnerability scan. Table 7-16 presents the comparison of the reported results with the results obtained by the clustering algorithms.

**Table 7-16 Comparison of the reported results with Clustering result**

<b>Type of Anomaly</b>	<b>No. of Occurrences</b>	<b>Hierarchical Clustering</b>	<b>DBSCAN</b>	<b>Sqalli et al, [19]</b>
Vulnerability Scanning	1	Yes, 1 Cluster	Not detected	Yes
phpmyadmin Attacks	1	Yes, 1 Cluster	Yes, 3 Cluster	Yes
Connection attempts on popular ports	1	Yes, 1 Cluster	Yes, 1 Cluster	Yes
SIP Worm	1	Yes, 1 Cluster	Yes, 1 Cluster	Not detected

### 7.3.4 Lab Capture

This trace was created inside the lab environment. The purpose of this is to create synthetic anomalies trace, and then apply the anomaly detection technique to check whether it will detect these anomalies. We also used the same lab capture trace to evaluate the performance of the clustering algorithms. Table 7-17 shows the reported results for the Lab capture trace.

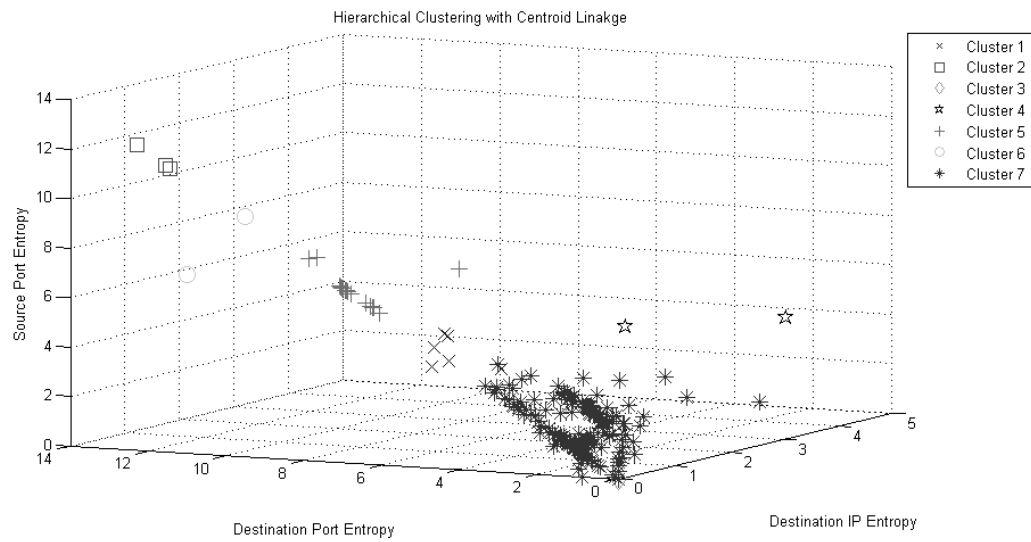
Table 7-17 Reported results for the Lab Capture trace.

Type of Anomaly	No. of Occurrences
System Compromise, ICMP Flood	3
System Compromise	12
Brute Force password guessing	4
SMB Attack	2
Port Scan- NMAP	5
Vulnerability Scan	1
SSH Attack	1

#### 7.3.4.1 Hierarchical Clustering

When we applied the Hierarchical clustering to the Lab capture trace, it produced seven clusters. Cluster 6 matches the threshold of the port scan attack. Cluster 2 represents the Vulnerability scan attack on the honeypot. Cluster 5 represents the ICMP Flood attack. Cluster 4 represents the system compromise attack on the SMB service. Cluster 7 matches the threshold for the system compromise attack. And, cluster 1

matches the threshold of the brute force password guessing attack, while cluster 3 represents the noise present in the dataset. The noise in the dataset is mostly due to the packets, which are not directed to or from the Honeypot IP, and this happens mostly in the switch level datasets. Figure 7-7 represents the 3-D plot of the Hierarchical clustering produced for the lab capture trace.



**Figure 7-7 3-D plot of the Hierarchical Clustering for the Lab Capture Trace**

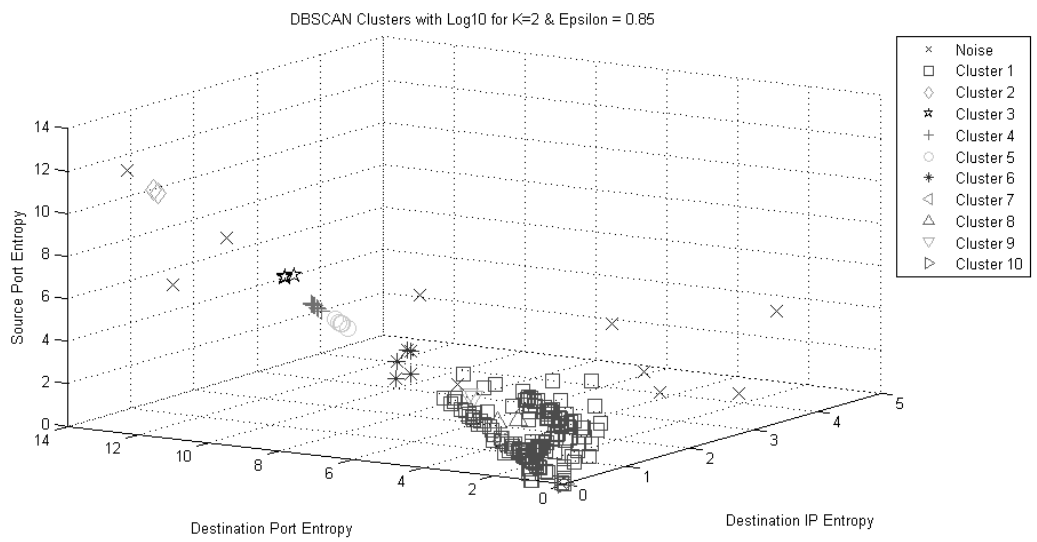
Table 7-18 shows the Hierarchical Clustering of the Lab Capture trace with the minimum and maximum values of each cluster.

**Table 7-18 Hierarchical Clustering of Lab Capture trace with Min-Max Values**

	Dst Port Ent		Src Port Ent		Dst IP Ent		Total Payload Bytes		Total Packets	
	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max
Cluster 1-BF	2.99	4.75	4.06	5.38	0	0.08	5	5.96	3	3.33
Cluster 2-VS	11.45	12.32	11.46	12.33	0	0.02	5.86	5.99	4.07	4.2
Cluster 3-noise	0	0	0	0	0	0	0	0	0	0
Cluster 4-SMB	1.89	4.09	3.99	4.14	3.06	4.42	3.89	4.47	2.05	2.13
Cluster 5-ICMP	4.05	8.07	6.13	8.12	0	0.23	4.86	6.21	3.07	4.33
Cluster 6-PS	9.62	11.03	7.23	9.64	0	0.07	5.1	5.96	3.31	4.16
Cluster 7-SC	0	3.92	0	3.92	0	3.37	2.39	5.67	0	3.3

#### 7.3.4.2 DBSCAN

DBSCAN Clustering of the lab capture trace produced ten clusters. Clusters 1, 7, and 8 match the threshold of the System Compromise attack. Clusters 3, 4, and 5 represent the system compromise attack by ICMP Flooding. Cluster 9 matches the threshold of the System Compromise attack on the SMB service. Cluster 6 represents the brute force password guessing attack. Cluster 2 represents the port scan attack. Cluster 10 represents the noise in the dataset. Figure 7-8 shows the 3-D plot of the DBSCAN clustering for the lab capture trace.



**Figure 7-8 3-D plot of DBSCAN clustering of Lab Capture Trace**

Table 7-19 shows the DBSCAN Clustering of the lab capture trace with the minimum and maximum values of each cluster.

**Table 7-19 DBSCAN Clustering of Lab Capture Trace with Min-Max Values**

	Dst Port Ent		Src Port Ent		Dst IP Ent		Total Payload Bytes		Total Packets	
	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max
Cluster 1-SC	0	3.92	0	3.92	0	1.94	2.39	4.88	0.6	2.85
Cluster 2-PS	11.45	11.57	11.46	11.58	0	0.01	5.86	5.99	4.07	4.2
Cluster 3 ICMP	8.02	8.07	8.05	8.09	0.11	0.23	4.86	4.88	3.07	3.09
Cluster 4- ICMP	6.83	7.12	6.83	7.12	0	0	5.4	5.46	3.49	3.55
Cluster 5-ICMP	6.09	6.44	6.13	6.49	0	0	6.09	6.21	4.24	4.33
Cluster 6-BF	4.33	4.75	4.06	5.38	0	0.08	5	5.53	3	3.33
Cluster 7-SC	1.25	1.69	1.26	1.72	0	0	5.42	5.67	3.06	3.3
Cluster 8-SC	1.86	2.17	2.19	2.59	0	0.49	5.18	5.47	2.38	2.51
Cluster 9-SMB	3.18	3.28	3.18	3.31	0.35	0.35	5.16	5.2	2.54	2.54
Cluster 10-Noise	0	0	0	0	0	0	0	0	0	0

Table 7-20 represents the comparison of the reported results with the results produced by the clustering process, and it also includes the comparison with the results reported by sqalli et al. [19]. Hierarchical clustering was able to identify all the attacks and put them in separate clusters. Only two attacks, SSH attack and path overflow exploit, are not detected by the Hierarchical clustering algorithm. Other than these two attacks, all other attacks were detected. DBSCAN also performed very well on this trace as it was able to



identify 93% of anomalies. For ICMP flood attack and System compromise attack, DBSCAN created three separate clusters for each attack. The reason for creating three separate clusters is the difference in the intensity of the attack, but all three clusters lay inside the defined threshold for the attack, so all the attacks are labeled under the same anomaly type.

**Table 7-20 Comparison of the reported results with Clustering result**

<b>Type of Anomaly</b>	<b>No. of Occurrences</b>	<b>Hierarchical Clustering</b>	<b>DBSCAN</b>	<b>Sqalli et al, [19]</b>
System Compromise, ICMP Flood	3	Yes, 1 cluster	Yes, 3 Clusters	Yes
System Compromise	12	Yes, 1 cluster	Yes, 3 Clusters	Yes
Brute Force password guessing	4	Yes, 1 cluster	Yes, 1 Cluster	Yes
SMB Attack	24	Yes, 1 cluster	Yes, 1 Cluster	Yes
Port Scan- NMAP	5	Yes, 1 cluster	Yes, 1 Cluster	Yes
Vulnerability Scan	1	Yes, 1 cluster	Not Detected	Yes
SSH Attack	1	Merged in other Cluster	Merged in other Cluster	No
Path Overflow exploit	1	Merged in other Cluster	Merged in other Cluster	No

## **7.4 RESULTS OVERVIEW**

Table 7-21 shows the performance overview of both clustering algorithms, Hierarchical Clustering and DBSCAN, against all the network trace files used for the experimentation. It also compares our results with the results reported by Sqalli et al. [19]. In Table 7-21, the first column represents the trace files used, the second column

represents the number of anomalies detected by Sqalli et al. [19], while the third and fourth columns represent the number of anomalies detected by Hierarchical clustering and DBSCAN, respectively, along with their percentage of success in comparison with the results reported by Sqalli et al. [19].

**Table 7-21 Performance of clustering algorithms against each Trace file**

Trace Name	Sqalli et al. [19]	Hierarchical Clustering	DBSCAN
Scan 27	17	14 (82%)	12 (71%)
Dionaea Capture trace-1	12	12 (100%)	11 (92%)
Dionaea Capture trace-2	5	5 (100%)	4 (80%)
Lab Capture	15	15 (100%)	14 (93%)

Table 7-22 shows the performance overview of both clustering algorithms, Hierarchical Clustering and DBSCAN, against different anomaly types present in the network trace files used for the experimentation. It also compares our results with the results reported by Sqalli et al. [19].

**Table 7-22 Performance of clustering algorithms against each Anomaly Type**

Anomaly Type	Sqalli et al. [19]	Hierarchical Clustering	DBSCAN
System Compromise	66	61 (92.4%)	62 (93.9%)
IRC Communication	2	2 (100%)	2 (100%)
Malicious File Download	1	1 (100%)	1 (100%)
Port Scan	13	13 (100%)	8 (61.5%)

Tables 7-21 and 7-22 present the performance results of Hierarchical clustering and DBSCAN clustering in comparison with the results reported by Sqalli et al. [19] along with the percentage success. The results calculated by the automatic clustering algorithms showed performance comparable to the anomaly detection technique used by Sqalli et al. [19], with the added advantage of automation of the whole anomaly detection process. In this way, we can analyze very large network trace files in very limited time by using the clustering algorithms. The detailed analysis of these results along with the future recommendations is discussed in the Chapter 8.

## **CHAPTER 8**

### **Conclusion and Future Work**

Honeypots are deployed to have a better understanding of the network attacks and network intrusions. So, analyzing the data produced by the Honeypots becomes a pivotal step in the understanding of the network attacks. We can easily relate the success of a Honeypot to the amount of data successfully analyzed and the network attack information extracted from the Honeypot data. The main focus of this thesis is to use an anomaly detection technique alongside with the automatic clustering of the detected anomalies. In this thesis, we used an anomaly detection technique which uses a combination of the volume and IP based features to calculate the entropy or volume values of the following selected features:

- Destination Port Entropy
- Source Port Entropy
- Destination IP Entropy
- Total Payload Bytes
- Total Packet Count

Once the entropy and log of volume values are calculated, these values are passed as input to the two clustering algorithms, i.e., DBSCAN and Hierarchical clustering. Both

clustering algorithms are implemented separately and their output is compared, to find out which clustering algorithm is a better candidate to use as an automatic clustering algorithm for anomaly detection in Honeynets.

Initially, we used some training datasets to check the suitability of the clustering algorithms and find the best combination of different tuning parameters. Initial experiments with the training datasets proved that the clusters created by the clustering algorithms are very much similar to the anomalies detected by the manual analysis of the entropy and volume values. Then, we applied both clustering algorithms to the unknown datasets and compared the results with the reported anomalies available in the datasets as well as those reported in earlier work.

Our first and major contribution for thesis work is to develop a framework for the automatic cluster creation for the known anomalies based on their threshold values for each traffic feature. To develop this framework, we implemented two different clustering algorithms, Hierarchical Clustering and DBSCAN Clustering. Both algorithms were successful in identifying different anomalies from the unknown datasets. Another contribution of our work is the improvement of the time required to detect anomalies for any new dataset. This also minimizes the human input in the process of analyzing dataset and detecting anomalies present in it. If we use Hierarchical Clustering then the whole process is automated, and no human input is required. While for using DBSCAN, the only human input required is in providing the tuning parameters, *Epsilon* and *Minimum Points*.

## 8.1 FUTURE WORK

The work presented in this thesis can be extended, so that it can provide more accurate and complete detection of different kinds of anomalies. Some ideas for future work are listed below.

- A large dataset of known anomalies can be created by using different training datasets. This will help us to increase the detection rate as well as the accuracy of the detected results.
- The antivirus companies and security analysts create defenses against new malwares by reverse engineering the network trace files. Our proposed anomaly detection mechanism can also be very helpful in pointing to some interesting segments in the network trace file, as each cluster member can lead directly to the time window where the actual attack has happened. This becomes very useful in case of very large trace files which are normally very difficult to reverse engineer because of the large number of packets as well as the long duration of the trace.
- The amount of information provided by each cluster can be increased by adding more feature information with each data element, i.e., time information. By using the time distribution and the placement of the data elements inside the clusters, we can detect multiple attacks of the same type inside a particular cluster. And this will also help to create more resilient attack signatures.

- In this thesis, we applied both clustering algorithms separately from each other. In future, we can look at the possibility of having these two clustering algorithms working in cascade or in some other possible combination, in such a way that the output of one clustering algorithm could become the input for the other clustering algorithm. This may help us in fine tuning the cluster boundaries.

## **8.2 LIMITATIONS**

The first drawback which we faced using the DBSCAN is its inability to detect the sparse anomalies. Sparse anomalies do not have the required minimum number of data points inside the neighborhood radius to create a DBSCAN cluster. In this case, DBSCAN does not create a cluster for the sparse anomaly, but instead it considers that particular anomaly as a noise.

The second drawback came in the use of Hierarchical clustering, as it successfully detected most of the anomalies but it was unable to differentiate between two attacks of the same type but different intensities, or two specific attacks under a generalized category. This inability is due to the natural clustering criteria used in the Hierarchical clustering, since natural clustering criteria only create clusters when the dissimilarity coefficient has the maximum value.

## References

- [1] L. Spitzner, *Honeypots: tracking hackers*: Addison-Wesley Professional, 2003.
- [2] L. Spitzner. (2012, April 10). *Honeypots: Definitions and Value of Honeypots*. Available: <http://www.spitzner.net/honeypots.html>
- [3] P. Barford, Y. Chen, A. Goyal, Z. Li, V. Paxson, and V. Yegneswaran, "Employing Honeynets For Network Situational Awareness," *Cyber Situational Awareness*, pp. 71-102, 2010.
- [4] D. Watson and J. Riden, "The honeynet project: Data collection tools, infrastructure, archives and analysis," in *Proceedings of the 2008 WOMBAT Workshop on Information Security Threats Data Collection and Sharing*, 2008, pp. 24-30.
- [5] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys (CSUR)*, vol. 41, p. 15, 2009.
- [6] P. J. Rousseeuw, A. M. Leroy, and J. Wiley, *Robust regression and outlier detection* vol. 3: Wiley Online Library, 1987.
- [7] M. Markou and S. Singh, "Novelty detection: a review--part 1: statistical approaches," *Signal Processing*, vol. 83, pp. 2481-2497, 2003.
- [8] P. N. Tan, M. Steinbach, and V. Kumar, *Introduction to data mining*: Pearson Addison Wesley Boston, 2006.
- [9] H. Jiawei and M. Kamber, "Data mining: concepts and techniques," *San Francisco, CA, itd: Morgan Kaufmann*, vol. 5, 2001.
- [10] F. Gong, "Deciphering detection techniques: Part ii anomaly-based intrusion detection," *White Paper, McAfee Security*, 2003.
- [11] A. Lakhina, M. Crovella, C. Diot, A. Lakhina, M. Crovella, and C. Diot, "Mining anomalies using traffic feature distributions," presented at the Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications, Philadelphia, Pennsylvania, USA, 2005.
- [12] G. Nychis, V. Sekar, D. G. Andersen, H. Kim, and H. Zhang, "An empirical evaluation of entropy-based traffic anomaly detection," presented at the Proceedings of the 8th ACM SIGCOMM conference on Internet measurement, Vouliagmeni, Greece, 2008.
- [13] P. Barford, J. Kline, D. Plonka, and A. Ron, "A signal analysis of network traffic anomalies," presented at the Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurement, Marseille, France, 2002.
- [14] A. Dainotti, A. Pescapé, and G. Ventre, "Wavelet-based Detection of DoS Attacks," in *Global Telecommunications Conference, 2006. GLOBECOM '06. IEEE*, 2006, pp. 1-6.



- [15] J. Haggerty, T. Berry, Q. Shi, and M. Merabti, "DiDDeM: a system for early detection of TCP SYN flood attacks," in *Global Telecommunications Conference, 2004. GLOBECOM '04. IEEE*, 2004, pp. 2037-2042 Vol.4.
- [16] D. Ping and S. Abe, "Detecting DoS attacks using packet size distribution," in *Bio-Inspired Models of Network, Information and Computing Systems, 2007. Bionetics 2007. 2nd*, 2007, pp. 93-96.
- [17] A. Kind, M. P. Stoecklin, and X. Dimitropoulos, "Histogram-based traffic anomaly detection," *Network and Service Management, IEEE Transactions on*, vol. 6, pp. 110-121, 2009.
- [18] F. Al-Haidari, M. Sqalli, K. Salah, and J. Hamodi, "An entropy-based countermeasure against intelligent DoS attacks targeting firewalls," in *Policies for Distributed Systems and Networks, 2009. POLICY 2009. IEEE International Symposium on*, 2009, pp. 41-44.
- [19] M. H. Sqalli, S. N. Firdous, Z. Baig, and F. Azzedin, "An Entropy and Volume-Based Approach for Identifying Malicious Activities in Honeynet Traffic," in *Cyberworlds (CW), 2011 International Conference on*, 2011, pp. 23-30.
- [20] D. Barbara, J. Couto, S. Jajodia, L. Popyack, and N. Wu, "ADAM: Detecting intrusions by data mining," in *In Proceedings of the IEEE Workshop on Information Assurance and Security*, 2001.
- [21] O. Thonnard and M. Dacier, "A framework for attack patterns' discovery in honeynet data," *digital investigation*, vol. 5, pp. S128-S139, 2008.
- [22] H. Jin, O. de Vel, K. Zhang, and N. Liu, "Knowledge discovery from honeypot data for monitoring malicious attacks," *AI 2008: Advances in Artificial Intelligence*, pp. 470-481, 2008.
- [23] T. Cover and P. Hart, "Nearest neighbor pattern classification," *Information Theory, IEEE Transactions on*, vol. 13, pp. 21-27, 1967.
- [24] M. M. Breunig, H. P. Kriegel, R. T. Ng, and J. Sander, "LOF: identifying density-based local outliers," *Sigmod Record*, vol. 29, pp. 93-104, 2000.
- [25] M. Ankerst, M. M. Breunig, H. P. Kriegel, and J. Sander, "OPTICS: ordering points to identify the clustering structure," *ACM SIGMOD Record*, vol. 28, pp. 49-60, 1999.
- [26] A. Ghourabi, T. Abbes, and A. Bouhoula, "Data analyzer based on data mining for Honeypot Router," presented at the Proceedings of the ACS/IEEE International Conference on Computer Systems and Applications - AICCSA 2010, 2010.
- [27] M. Ester, H. P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *A density-based algorithm for discovering clusters in large spatial databases with noise*, 1996, pp. 226-231.
- [28] D. Fisher, "Improving inference through conceptual clustering," in *Proceedings of the sixth National conference on Artificial intelligence - Volume 2*, 1987, pp. 461-465.
- [29] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, 1967, p. 14.
- [30] A. Maheshwari and M. M. Dabbeeru, "Title," unpublished].

- [31] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to information retrieval* vol. 1: Cambridge University Press Cambridge, 2008.
- [32] N. K. Malhotra, *Marketing Research: An Applied Orientation*, 5/e: Pearson Education India, 2008.
- [33] (2010, June 5). *Honeynet.org. Honeynet Project, Honeynet Definitions, Requirements, and Standards Documentation, Honeynet Project website.* Available: <http://old.honeynet.org/alliance/requirements.html>
- [34] Fyodor. (2011, July 25). *Top 100 Network Security Tools* Available: [sectools.org/](http://sectools.org/)
- [35] P. Laskov and M. Kloft, "A framework for quantitative security analysis of machine learning," in *Proceedings of the 2nd ACM workshop on Security and artificial intelligence*, 2009, pp. 1-4.
- [36] K. Rieck and P. Laskov, "Language models for detection of unknown attacks in network traffic," *Journal in Computer Virology*, vol. 2, pp. 243-256, 2007.
- [37] P. Düssel, C. Gehl, P. Laskov, J. U. Bußer, C. Störmann, and J. Kästner, "Cyber-critical infrastructure protection using real-time payload-based anomaly detection," *Critical Information Infrastructures Security*, pp. 85-97, 2010.

## VITAE

- Muhammad Shoieb Arshad
- Born in Bahawalpur, Pakistan, on 16<sup>th</sup> October, 1985
- Nationality: Pakistani
- Received B.S Computer Engineering from COMSATS Institute of Information Technology (CIIT) in August 2007
- Worked in COMSATS Institute of Information Technology (CIIT) as Lecturer from Nov,2007 to Feb,2010
- Joined King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia as a Research Assistant in February, 2009.
- Completed M.S in Computer Networks from King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia in May 2012.
- E-mail: [shoieb.arshad@gmail.com](mailto:shoieb.arshad@gmail.com)
- Phone: 00966546975870
- Present Address: P.O. Box 1542, King Fahd University of Petroleum and Minerals, Dhahran-31261, Saudi Arabia
- Permanent Address: H. No. 29, Muslim Town, Bahawalpur, Pakistan