THE REPUBLIC OF TURKEY
BAHCESEHIR UNIVERSITY

AN APPROACH FOR CLASSIFYING ALERTS OF INTRUSION DETECTION SYSTEMS

Master's Thesis

FARSHID POURABBAS

ISTANBUL, 2014

THE REPUBLIC OF TURKEY BAHCESEHIR UNIVERSITY

GRADUATE SCHOOL OF
INFORMATION TECHNOLOGIES

AN APPROACH FOR CLASSIFYING ALERTS OF INTRUSION DETECTION SYSTEMS

Master's Thesis

FARSHID POURABBAS

Supervisor: Prof. Dr. Adem KARAHOCA

ISTANBUL, 2014

THE REPUBLIC OF TURKEY BAHCESEHIR UNIVERSITY

INSTITUTE OF SCIENCES

Name of the thesis: AN APPROACH FOR CLASSIFYING ALERTS OF INTRUSION DETECTION SYSTEMS

Name of the Student: Farshid Pourabbas

Date of thesis defense: 15 January 2014

The thesis has been approved by the Institute of Sciences.

Assoc.Prof.Dr. M.Tunç Bozbura

Graduate School Director

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

Examining Committee Members	Signature
Prof.Dr. Adem Karahoca (Supervisor)	
Assoc.Prof.Dr. M.Alper Tunga	
Asst.Prof.Dr. Yalçın Çekiç	

ACKNOWLEDGEMENTS

I express sincere appreciation to my thesis advisor Prof. Dr. Adem KARAHOCA for guiding and facilitating my research activities.

Date: 15.01.2014

Farshid Pourabbas

ÖZET

SALDIRI TESPİT SİSTEMİNİN UYARILARINI SINIFLANDIRAN BİR YAKLAŞIM

Farshid Pourabbas

Bilgi Teknolojileri Yüksek Lisans Programı

Tez Danışmanı: Prof. Dr. ADEM KARAHOCA

Ocak 2014, 66 Sayfa

Internet ağlarının büyümesi ile bugün, veri alışverişi güvenliği önemli bir görev olarak kabul edilmektedir. Bu nedenle, güvenlik araçlarının kullanımı gün geçtikçe artmaktadır. Saldırı tespit sistemleri bu araçlar arasında yer alıyor. Uyarı olarak bir ağdan alınan mesajı etiketlemek mümkündür, ancak sistem durumunu açıklamak mümkün değildir. Bazı yöntemler saldırı tespit sistemlerinden alınan uyarıların ilişkilendirilmesi yoluyla yukarıdaki sorunu çözmek için geliştirilmiştir. Uyarıları birbirleriyle ilişkilendiren yöntemlerle, sistem durumunu açıklamak mümkün olacaktır. Uyarıların korelasyon yöntemlerinin adımlarından biri de sınıflandırmadır. Sınıflandırma verimli bir şekilde gerçekleştirildiğinde daha iyi bir sistem durumu tanımlanabilir. Burada, uyarıları sınıflandırmak için bir yöntem geliştirilmiştir.

Anahtar Kelimeler: Saldırı Tespit Sistemleri, Uyarılar, Sınıflama, ilişki, Veri Madenciliği

ABSTRACT

AN APPROACH FOR CLASSIFYING ALERTS OF INTRUSION DETECTION SYSTEMS

Farshid Pourabbas

Master of Science in Information Technologies

Supervisor: Prof. Dr. ADEM KARAHOCA

January 2014, 66 Pages

With the growth of the Internet networks today, security of data exchange is considered as an important task. Therefore, the use of security tools is increasing day by day. Intrusion detection systems are among these tools. They are only able to label a message received from a network as ‘alert’, but they are unable to describe system status. Some methods have been developed to solve the above problem through correlating the alerts received from intrusion detection systems. By correlating the interrelated alerts, the methods would be able to describe system status. One of the steps of correlation methods of alerts is to classify them. System status can be described well when classification is performed efficiently. Here, we present a method for classifying alerts.

Keywords: Intrusion Detection Systems, Alerts, Classification, Correlation, Data Mining

CONTENTS

TABLES

FIGURES

ABBREVIATIONS

ACM	:	Alert Correlation Matrix
CBE	:	Common Base Event
DARPA	:	Defense Advanced Research Projects Agency
DFS	:	Depth First Search
DSS	:	Digital Signature Standard
DTD	:	Document Type Definition
GCT	:	Granger Causality Test
GLA	:	Generic Log Adapter
IDMEF	:	Intrusion Detection Message Exchange Format
IDWG	:	Intrusion Detection Working Group
IDS	:	Intrusion Detection Systems
IETF	:	Internet Engineering Task Force
IP	:	Internet Protocol
MLP	:	Multi-Layer perception
NB	:	Native Bayesian
NIDS	:	Network Intrusion Detection Systems
NMF	:	Network Management Field
NMS	:	Network Management System
NTP	:	Network Time Protocol
SCADA	:	Industrial Processes Control Systems
SLA	:	Service Level Agreement
SVM	:	Support Vector Machine
XML	:	Extensible Markup Language

SYMBOLS

Maximum order of identical bits	:	n
Correlation between two alerts in the alert correlation matrix	:	$W_{C\ (ai,\ aj)}$
An attack graph as a directed graph	:	G
Set of vertices representing the alert	:	V
Set of edges (v_i, v_j) indicating transfer from alert a_i to alert a_j	:	E
The set $c_{i,\ j}$ indicating correlation strength between two alerts	:	C
Queue	:	q
List of alerts	:	A
Correlation threshold	:	r
Correlation sensitivity	:	s
Hyper-alert list	:	H

1. INTRODUCTION

Today, with the numerous attacks and sabotages occurring over networks and threatening performance of many customers and its users, security centers attempted to look for solutions to maintain security over the network more than ever. Various security tools, such as firewalls, intrusion detection systems, etc. are used to improve security level on a network.

One of the major problems of intrusion detection systems is issuance of many alerts with low-level abstraction. To solve this problem, we need to have some methods to issue alerts with higher abstraction level while reducing alerts and removing wrong alerts.

With respect to the very large volume of data passing over the network, importance and confidentiality of the data, necessity to maintain security and protect users' data in today's world, there is a pressing need to have a security system to be able to manage network and protect system against possible damages.

In typical systems, the tools such as firewalls, antivirus software, and intrusion detection systems attempt to protect a network and defend against possible attacks. These tools are suitable solutions to reduce the impacts of computer attacks; however, they cannot be considered as an inclusive approach to protect and prevent network form possible damages. One of the tools that gained attentions recently is intrusion detection systems. They are able to detect and issue an alert. Two problems concerning intrusion detection systems are "A large number of received alerts" and "Wrong alerts".

With respect to the above items and the fact that the issued alerts have low abstraction levels, as a network manager would have no understanding of system status, we need a system to enable us to detect relationships between these alerts. This is realized by collecting alerts from intrusion detection systems and providing network manager with a high-level vision for the attacks taken place over the network.

‘Correlation of alerts’ means establishing a relationship between some alerts and promoting them to higher-level alerts whose management is more convenient for network manager.

As data mining means extraction of useful information out of a large volume of data, it can be used as a method for correlating alerts. Some of the models proposed for correlating alerts enjoy data mining techniques; however, all these methods have some drawbacks that make us keep looking for some efficient methods with low computational and memory overhead.

The rest of the article discusses the following items: Section two reviews literature, the proposed method is explained in section three, conclusions are brought about in section four, and section five discusses the results.

1.1 PROBLEM STATEMENT

Nowadays, numerous attacks and sabotages occurring in the network threatened many customers and users of the network; this issue has made the security companies look for ways, more than before, to protect the security of their networks. In order to enhance the security level of the network, they use various security tools such as intrusion detection systems, firewalls, and anti-viruses.

Expansion of the network and the security devices always cause a lot of alerts from the network elements to be sent to the network administrator. Unfortunately, the number of these alarms is so high, and their abstraction level is so low that their analysis is not practically possible for the network administrators. In addition, many of these alerts are false positives.

Intrusion detection systems are one of the available methods for monitoring the security status of the network and their analysis.

Intrusion detection systems are an appropriate solution for reducing the impact of computer attacks, but why they cannot be considered as a comprehensive solution for protecting and preventing from possible harms for the network. There are many reasons for this, which are noted in the following examples:

i. The high number of received alerts: the first case is the high number of alerts generated by these tools, such that alert numbers received in a computer network may be about 15,000 alerts per second. Managing these alerts for a human user, who is supposed to monitor and manage the system, is very difficult and practically impossible.

ii. False alerts: false alerts caused the decision-making not to be done properly in connection with the attacks on the network. These false alerts may exist in two forms. The first group is false positive alerts, including alerts generated wrongly. For example, suppose that Linux operating system has been installed on a victim system, and the attacker performs an attack which is capable of affecting the Windows operating system, if intrusion detection systems issue an alert, this alert certainly will be a false positive alert. The second group is false negative alerts, including alerts which must be generated by intrusion detection systems, but they have not been generated. These alerts are detectable by examining the previous and the next alerts.

iii. Incapability of discovering complex attacks: the biggest problem of tools such as intrusion detection systems and firewalls is that they are incapable of detecting complex and multi-stage attacks. Nowadays cyber threats are not summarized in simple and one-step attacks, and attackers, by working on the vulnerabilities of target systems, are trying to break the security system’s lock in some interconnected and interrelated stages, which cause the intrusion detection systems not to find the communication between various stages and to reveal behind the curtain of scenario.

Considering the above reasons, we need a system that can find the relationship between these alerts, by collecting alerts from intrusion detection systems, to provide a high vision of network attacks for human user and the network monitoring system.

Correlation of alerts means establishing a relationship between several alerts, and upgrading them to high level alerts that their management is easier for the network administrators.

Since data mining means extracting useful information from a massive amount of data, so it can be used as a method for correlating alerts. Some of the proposed methods of correlating alerts have utilized data mining techniques, but all of these methods have some weak points that still make us to provide efficient methods with low computational and memory overhead.

In this thesis, in addition to reviewing the previous works in the field of correlating alerts and their analysis, we propose a method based on data mining (frequent pattern mining) for correlating alerts.

1.2 BACKGROUND

The aim of correlation is providing a comprehensive outlook of events occurring in the system for the system administers or the system replying to hack. Instead of generating hundreds of discrete and low-level alerts, the correlator presents a high-level alerts or a scenario to the system administer, by verifying the generated alerts and discovering their logical relations.

Correlating is also known as construction of attack scenario or extraction of attack scenario. Works performed in the field of alerts correlation are divided into the following categories, based on how they correlate the alerts:

1.2.1 Alert Correlation Cased on a Known Scenario:

In these methods, causal relationships between alerts are expressed as attack scenario. Two alerts are correlated with each other, if the possibility of constructing of attack scenario exists with their combination. The main problem of this category of methods is that if the attacker uses a new scenario for attack, these methods will not be able to correlate the alerts. The most important languages that are used in these methods to describe attacks include:
STATL, LAMBDA, ADELE

Attack scenario can be achieved by two methods:

a) Using previous knowledge
b) Using machine learning techniques.

Authors have used the method of prior knowledge to obtain attack scenarios. To fix the problem of slow attacks they have used queue graph to correlate the alerts. Queue graph is an attack graph, in which they put a queue instead of each exploit, and they put a variable instead of each security condition. Attack graph consists of a set of security conditions and exploits (Wang et al., 2006).

Only one alert can be placed in each queue. When an alert is given, its corresponding exploit is specified and placed in the queue by using a function. In this function, the mapping is performed by using prior knowledge.

Identifying all the attack scenarios by using prior knowledge requires so much time and labor. These methods are not able to detect new attacks. To overcome these problems, machine learning techniques and data mining techniques used to extract the relationship between alarms. By using MLP and support vector machine classifier, the probability of correlation between two alerts based on the similarities among features of IP source, IP destination, destination port, and time stamp of alert is calculated (Zhu and Ghorbani, 2006).

In this method, when a new alert arrives, at first, a hyper alert possessing the alert which has the highest probability of correlating with new alert is defined by using MLP neural network and support vector machine classifier. If the probability of the obtained correlation was lower than the correlation threshold, the new alert would not be correlated with any alert. If the calculated probability was higher than the threshold, then correlation probability of new alert would be calculated by all of present alerts in the obtained hyper alert.

Therefore, an alert would be correlated with the new alert that the difference between their probability and the highest probability, that previously were found, would be lower than the correlation sensitivity measure. If no alert was found for correlation, the new alert would be place in a new hyper alert.

AprioriAll algorithm has been used to correlate the alerts. AprioriAll is a sequential pattern mining algorithm. First, this algorithm is applied on a collection of alerts, and some collections of alert classes are generated as the output. Then these collections are converted into some graphs in which each node represents an attack class (Agrawal and Srikant, 2000).

1.2.2 Alert Correlation Based on Rules

Methods based on rules are one of the most significant methods in correlation between alerts, and many studies have been done on these methods. The idea of these methods is based on a principle that the attacks are not usually separated. Each attack provides the conditions for the next attack or the next stage of an attack. The relations between alerts in these methods are determined by applying the rules. Alert A is correlated with alert B, if alert A, based on existing rules, is prerequisite for alert B.

In Xu and Ning, the concept of resource is used for indicating the prerequisite and the consequences of an attack. The resource may be a port, a service, and so on. The prerequisite of an attack is named input resource, and the consequence of an attack is named output resource. In this methods the causal relationships between resources is provided as rules, and they are used in the correlation between alerts. We used partial adaptation in this article; it means that if the result of an alert meets at least one of the prerequisites of the other alert, those two alerts would be correlated with each other's (Xu and Ning, 2005).

A language for describing the attacks has been introduced in "*Building scenarios from a heterogeneous alert stream*". The concept of capability has been used in this language for expressing the prerequisites and consequences of attacks. Occurrence of each attack needs a series of capabilities, and this occurrence leads to the realization of a series of capabilities. The prediction of the next steps is possible in this method. In this method, all prerequisites of an attack must be satisfied, so that the attack would be considered. So one of the steps of a multi-step attack is not diagnosed by IDS, the attack scenario will never be detected (Dain and Cunningham, 2002).

In “*Constructing attack scenarios through correlation of intrusion alerts*”, “*Ning et al*” have paid attention to modeling of prerequisite relationship between alerts. Writers of this article have introduced a concept named hyper alert type. A hyper alert type is <fact, prerequisite, and consequence >.

Fact: a set of features including information about the alerts.
Prerequisite: a set of conditions required for the attack to be successful, which is described as a conjunction of a logical predicate.
Consequence: The results of an attack which is described in the form of a collection of logical predicates.

In this method, for correlating alerts, they are processed by a processor, in order to make hyper alerts. Then prerequisites of a hyper alert would be compared to the results of other hyper alerts, and if they are in accordance with each other, they would be correlated.

In this method, a graph named correlation graph is used for displaying correlation results. In this graph each node is a hyper alert and each edge represents the correlation between them (Ning, Cui and Reeves, 2002).

In another article by Ning “*Building attack scenarios through integration of complementary alert correlation method*”, a method is proposed for hypothesizing the missing alerts of intrusion detection systems. Authors of this article have combined methods based on similarity, prerequisite, and consequence, in order to recognize the attack scenarios in a high level manner (Ning, Xu, Christopher, Healey and Amant, 2004).

Performance of rule-based methods depends on the prior knowledge and skill of the expert. These methods do not work well when facing new attacks.

1.2.3 Alert correlation based on statistic

In these methods, if two alerts statistically depend on each other, they would correlate to each other. This category of methods requires a training database for mining attack scenarios. Thus, the efficiency of this category of methods depends on the amount of the credit of the database. Updating these methods is difficult. Because adding a new attack pattern needs retraining the system.

In "*A scalable continuous query system for internet databases*" a Bayesian network is used for modeling the causal relationships between alerts. In this method alerts are shown by nodes, and causal relations are shown by edges (Chen, DeWitt, Tian and Wang, 2000)

In another similar article, *Ren et al* have used Bayesian networks to extract information about causal relationship between alerts, and then reconstruct attack scenarios based on extracted information (Ren, Stakhanova and Ghorbani, 2010)

1.2.4 Temporal correlation

In these methods, two alerts are correlated to each other based on temporal relationships. Analysis of time series has been used to explore the causal relationship. In this method, alerts whose all features, except the temporal label, are the same is placed in one category.
Therefore a hyper alert is made for each category, based on the time of occurrence. In fact each hyper alert is a graph in which the alerts of a category have been connected together other in a chronological order. Then each hyper alert is divided into time intervals, and the number of alerts of the hyper alert in these intervals is determined. The collection of these numbers is a time series for each hyper alert (Qin and lee, 2003).

Lee and Keane have introduced a method in which causal relationships between alerts is analyzed by using Granger causality test (GCT). Granger causality test is a statistical analysis method based on time series that provide the possibility of analyzing the correlation of time series variable X with time series variable Y, by using statistical hypothesis testing. Hyper alerts are made for analyzing time series of correlation of alerts, by integration of alerts which all values of all features, except time differences, are the same (Granger, 1999).

Time series variables are modeled for all hyper alerts base on the number of alerts in the unit of time, and then the Granger causality test is performed. Granger causality test determines whether the variable X provides significant information about variable Y or not. If the answer is yes, then alert X could be the cause of the alert Y. This method is not highly dependent on prior knowledge (Reza Sadoddin, 2006).

As it was mentioned, one of the methods appropriate for correlating alerts is the use of machine learning and data mining techniques. One of the data mining methods is the frequent pattern mining. Based on this fact that most of attacks have frequent sequential features, the issue of behavioral pattern mining of attack can be converted into frequent pattern mining of alerts. Frequent patterns are patterns which have been repeated in a data set with a frequency more than or equal to what is determined by the user as a threshold. For example, some sets such as milk and bread, which are usually applied together in transactions of a transaction database in a store, are a set of frequent items.

Sadoddin et al have presented a framework for real-time correlation of alerts based on the number of simultaneous occurrence of alerts. The presented framework provided a tool for processing a continuous stream of alerts. They have extracted frequent patterns using Fp-Growth algorithm, and then they create attack scenarios (Sadoddin, 2006).

First in this method, alerts correlated to the graph structures, based on information of their connection considering alerts' source and destination. Each of these structural patterns may reveal attack strategies, or normal models, which have been created by false positive alerts. Then the FP-Growth algorithm has been used for mining the sequential structures. FP-Growth algorithm uses FP_Tree, which provides a compact data structure for storing frequent candidate patterns. This method has some problems. According to the authors, creating an FP_Tree needs a lot of memory and if the tree gets larger, processing time may be reduced.

In this thesis we are going to provide a method for correlating alerts, so as to reduce the number of alerts, and also to produce alerts with a higher level of abstraction. In this thesis we will try to provide a method for correlating alerts received from intrusion detection system, by using the methods of frequent pattern mining, and to extract attack scenarios.

1.3 RESEARCH OBJECTIVES

1.3.1 Scientific Objectives: In this thesis, we are trying to review the existing methods for correlating alerts received from intrusion detection systems, and to provide a new method.

1.3.2 Practical Objectives: in this thesis, we are trying to correlate the alerts received from IDS, and to extract attack scenarios.

1.3.3 Specific Requirements of Research: Expansion of the network and the security devices always cause a lot of alerts from the network elements to be sent to the network administrator. Unfortunately, the number of these alarms is so high, and their abstraction level is so low that their analysis is not practically possible for the network administrators. In addition, many of these alarms are false positives. In this thesis we are trying to correlate the alerts in order to generate alerts with a higher level of abstraction.

2. MATERIALS AND METHODS

2.1 CORRELATION METHODS ANALYSIS

First of all two alert correlation methods are described and their weaknesses and strengths are explained. Second, a combination of the two methods is used to develop a new model:

Alert correlation methods are classified into several groups. One of them uses machine learning techniques. The Zhu and Ghorbani method, which is described below, is an example of this type of methods. The advantage of these methods is that they do not require “formulated prior knowledge”.

2.2 ZHU-GHORBANI METHOD

The method introduced by *Ghorbani and Zhu (2006)* is classified into the group of methods based on machine learning. In this method, the correlation between two alerts is analyzed using neural networks, which act as a learner machine. Afterwards, the alerts are classified so that each group of alerts, known as a hyper-alert, represents a group of correlated alerts.

This method utilizes the *DARPA 2000* data set. This data set includes alerts generated by intrusion detection systems and saved as dump files. This method works by reading from a data set. The data is then prepared for process (because the data set contains raw data). Hence, preprocessing is necessary before implementing the method.

Zhu and Ghorbani diagnose correlations based on 6 qualities of correlation, which are as follows:

a) Similarity Between Source IP Addresses Of Two Alerts:

The source IP address of the alert reflects the identity of the attacker. Therefore, two attacks with one IP address are probably associated with one attack scenario and can be correlated. Similarities between two source IP addresses are calculated as follows:

$$\text{Sim (IP1, IP2)} = {}^{n}\!/_{32}$$

Source: B. Zhu, A. Ghorbani, (2006). *Alert correlation for extracting attack strategies.* International Journal of Network Security, vol. 3, no. 3, pp. 244–258.

In the above relation, n stands for the maximum sequence of bits equal in quantity and 32 is the length of an IP address. For example, the similarity value for the following two IP addresses is equal to 0.75:

IP1=192.168.0.001=> 11000000 10101000 00000000 00000001

IP2=192.168.0.201=> 11000000 10101000 00000000 10000001

Source: B. Zhu, A. Ghorbani, (2006). *Alert correlation for extracting attack strategies.* International Journal of Network Security, vol. 3, no. 3, pp. 244–258.

b) Similarity Between Destination IP Addresses Of Two Alerts:

The significance of the similarity between destination addresses is way more than the similarity between source addresses. The reason is that probability of correlativity of two alerts with two different destination addresses is very low. In order to obtain the similarity between two destination IP addresses relation (1) is used.

c) Identical Destination Port Numbers:

Before an attacker can exploit the vulnerability of a service that is listening on a certain port, he should know whether the port is open. Hence, this is very important for correlation between two alerts.

d) Equality Of Current Alert Source IP Address And Previous Alert Destination IP Address:

The reason for utilizing this capability is that attackers sometimes launch their attacks from another computer.

e) Rollback Correlation Between Two Alerts:

The value of this quality is between 0 and 1 and is obtained using the following relation.

$$\pi^{b}_{c(a_i,a_j)} = W_{c(a_i,a_j)} \Big/ \sum_{k=1}^{N} W_{c(a_i,a_j)}$$

Source: B. Zhu, A. Ghorbani, (2006). *Alert correlation for extracting attack strategies.* International Journal of Network Security, vol. 3, no. 3, pp. 244–258.

The term $W_{c(a_i,a_j)}$ in the above relation shows the correlation between two alerts in the Alert Correlation Matrix (ACM). In the following is described how the matrix is calculated.

f) Frequency Of Correlations Between Two Alerts:

The value of this quality is between 0 and 1. Values approaching zero indicate that the two alerts are rarely correlated. However, values close to one indicate that the two alerts are frequently correlated.

After extracting the above qualities, it's time for identifying correlations between two alerts. In this method, correlations are identified using the MLP model, which is a model of neural networks.

2.2.1 The MLP Model for Identifying Correlation between Two Alerts

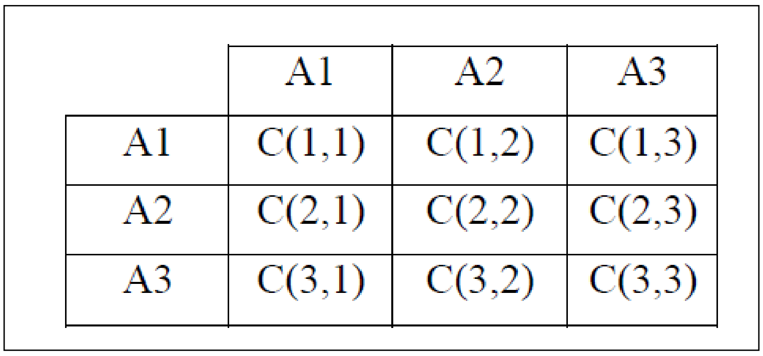
The neural networks employed here is a three-layer model with 6 neurons in the first layer, 7 neurons in the middle layer, and 1 neuron in the last (outer) layer. First an explanation of the network is required. We use data presented in the above mentioned source and use the network for analyzing the correlation between two alerts. That is to say, the aforementioned 6 qualities of the two alerts under study are calculated first and the values are used as the input for the neural network. The network output is a value that reflects the probability of existence of a correlation between the two alerts. The value is also used for creating the alert correlation matrix.

2.2.2 Alert Correlation Matrix (ACM)

The value (power) of the correlation between two alerts plays an important role in the analysis of attack patterns. It can reveal the causal relationships between two alerts. However, deciding on the value of correlation (or determining the power of the relationship between two alerts) is difficult since it requires wide knowledge of attacks and relations among them.

Zhu and Ghorbani used the Alert Correlation Coefficient, which shows the weights of correlations between two alerts. The power of a correlation can be calculated based on this matrix. Hence, it provides more information on methods for correlating alerts and analyzing attack strategies. The matrix is used with an aim to create attack graphs that help to discover the strategies implemented by attackers. The following figure shows the alert correlation matrix for three alerts. The values of elements show the probability of correlation between two alerts that can be obtained by the correlation engine.

Table 2.1: Alert Correlation Matrix (Acm)



	A1	A2	A3
A1	C(1,1)	C(1,2)	C(1,3)
A2	C(2,1)	C(2,2)	C(2,3)
A3	C(3,1)	C(3,2)	C(3,3)

Source: B. Zhu, A. Ghorbani, (2006). *Alert correlation*

2.2.3 Correlation Method

The output of the correlation engine is used for determining correlativity of two alerts. In the next step, a group of correlated alerts are shown. The group is named the hyper-alert group. In the following is presented an algorithm for the creation of hyper-alerts. However, first some definitions are required to be presented:

2.2.3.1 Hyper-alert: A hyper-alert is shown with the <V, E, C> ternary. V is the known as the vertices and shows the input alerts. E includes the edges and shows the relationship between two alerts. Finally, C shows the probability of correlativity of two alerts.

Each hyper-alert includes a number of interconnected alerts that mark a type of attack. In order to create a hyper-alert two thresholds are defined: correlation threshold and sensitivity threshold.

Correlation threshold is used for proving correlativity of two alerts. That is to say, the output of the correlation engine for the two alerts is compared with the correlation threshold. If the output exceeds the threshold the two alerts are known as correlated alerts; otherwise, they are considered uncorrelated alerts. In this method, the value of the correlation threshold is assumed to be 0.5.

Normally when an alert is included in a hyper-alert, it connects to the last alert in the hyper-alert. However, when an alert is a sequence of several other alerts it should be connected to several alerts in the hyper-alert. The sensitivity threshold determines the alert in the hyper-alert that should be connected to the input alert. To this end, the difference between the probability between two alerts and the probability between the received alert and other alerts in the hyper-alert is calculated. If the difference is less than the sensitivity alert, the incoming alert is connected to them. Here, the threshold is assumed to be 0.1.

Figure 2.1: Algorithm, Correlation Method

```
A: List of alerts
r: Correlation threshold
s: Correlation sensitivity
initialize hyper-alert list H
for all each alert ai in A
for all hyper-alert hj that contains an alert aj
such that the correlation probability of ai and aj is maximum
m ← this maximum correlation probability
if m > r then
for each alert ak in hj
if m-(probability between ak and ai) < s then
connect ai with ak
else
create a new hyper-alert
put ai in new hyper-alert
```

Source: B. Zhu, A. Ghorbani, (2006). *Alert correlation for extracting attack strategies.* International Journal of Network Security, vol. 3, no. 3, pp. 244–258.

2.2.4 Creating the Attack Graph Using the Alert Correlation Matrix

In this method ACM is used for creating the attack graph. The attack graph is created based on model data and output of the alert correlating algorithm. A typical attack graph is defined as follows:

An attack graph is an oriented graph that is shown as G= (V, E, C). V stands for the word Vertices and refers to alert a. E stands for the word Edge and includes (vi, vj). It reflects the transition from alert ai to alert aj. Finally, C includes the ci,j collection, which shows the power of correlation between alerts ai and aj. The process of creation of an attack graph using ACM is illustrated in the following algorithm (Zhu and Ghorbani, 2006)

Figure 2.2: Algorithm, ConstrucGraphsFormACM(a,r)

```
a: the starting alert of the attack graph
r: the $\pi^f$ threshold
initialize graph G
initialize queue q
q ←$a_i$
G ←$a_i$
While not q.isEmpty()
a ← q.dequeue()
for j ← 0 to number of alerts in ACM
if $\pi^f_{cell(a,a_j)} > r$
if $a_j$ has not been visited
q ←$a_j$
visit $a_j$
G ← G U (a,$a_j$)
$C_{i,j}$ ← $\pi^f_{cell(a,a_j)}$
G ← G U $C_{i,j}$
Return (G)
```

Source: B. Zhu, A. Ghorbani, (2006). *Alert correlation for extracting attack strategies.* International Journal of Network Security, vol. 3, no. 3, pp. 244–258.

2.2.5 Pros and Cons Of The Zhu-Ghorbani Method

The advantage of this method is that it does not required prior knowledge.

The disadvantage of this method is the relationships among alerts in a hyper-alert. It seems that the value of the sensitivity threshold is not appropriate because in practice only a few alerts are included in a hyper-alert.

2.3 LEE METHOD

Statistical methods are another means of correlating alerts. In these methods, Bayesian networks are commonly used for obtaining information on causal relationships among alerts. In the following a method is studied which uses Bayesian networks for obtaining information on relationships and regenerating attack scenarios.

2.3.1 Alert Correlation

In this method, first the alerts are aggregated and classified. Afterwards, classified alerts are prioritized prior to be subjected to correlation. Aggregation and classification of alerts lead to a reduction in redundancy of alerts. However, the important qualities of the alerts such as IP addresses, port number, etc. are retrained.

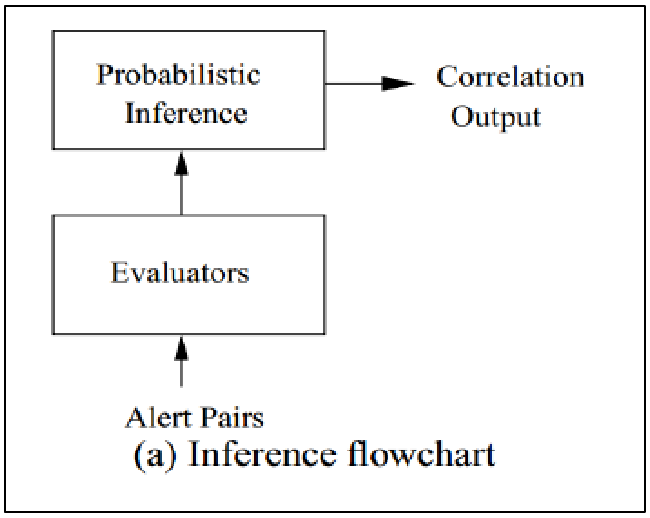
In this stage, received alerts of one single attack are aggregated. Aggregated alerts with identical qualities are classified into one group named the hyper-alert. Alert prioritization assigns orders to hyper-alerts based on the network configuration.

2.3.2 Probabilistic Inference For Alert Correlation

Experiments indicate that when a system is attacked it is either turned into a target for future attacks or is used as a platform for attacks to other systems. Hence, a sequence of attacks to a specific host can be used as a proof for other probable attacks. Although it is possible to correlate these attacks using if-then sentences, it is not possible to express all attacks using this structure. Therefore, here a probabilistic inference model is used for correlating alerts. It is done by establishing an indicator system of a sequence of attacks and having prior knowledge of transition of attacks. In the following is described how probabilistic inference can be used for a sequence of attacks to discover complex relationships among attacks.

The following figure shows the correlation inference process. At the beginning, the evaluator analyzes one or several qualities of a sequence of alerts and sends the result to the inference model. Next, the inference model turns the results of evaluations by evaluators into a decision through calculation and distribution of correlation beliefs over the inference model (Bayesian network).

Figure 2.3: Inference flowchart



Source: B. Zhu, A. Ghorbani, (2006). *Alert correlation*

The above procedure includes the following steps:

In the first step the alerts are accumulate and classified. The objective of this phase is to reduce redundancy among alerts. Alerts are accumulated based on their properties such as source IP address, destination IP address, destination port, etc. Next, alerts with identical properties are classified into one group and each group is called a hyper-alert.

In the second step the Bayesian network is used for explaining relations between alerts. These relations are derived from prior knowledge.

2.3.3 Bayesian Network

The Bayesian network is a directed acyclic graph in which vertices show variables and edges show the relationships among them. In this network relationships are expressed as conditional probabilities. The Bayesian network has two parameters to be learnt. Learning in a Bayesian network refers to the process of learning these two parameters. The first parameter includes the probability of nodes without parents (the so called prior nodes). The second parameter is allocated to the conditional probabilities between parents and children (which are called evidence).

The output from the Bayesian network depends on the precision of these two parameters. Hence, these parameters should be determined based on prior knowledge (Xu and Ning, 2005).

2.2.4 Pros and Cons of the Lee Method

a) The advantage of the method is that it explains the relationship between alerts based on useful statistical information.

b) The disadvantage of this method is that it requires prior knowledge.

2.4 RECOMMENDED METHOD

It seems that the two methods discussed above are successful in correlating alerts. In the following we are going to explain our plan using the two mentioned methods.

The Alert Correlation Method (ACM) is useful for demonstrating correlation between two alerts and extracting an attack scenario. However, the relationship between alerts contained in a hyper-alert is not satisfactory. Hence, we are going to introduce a method for relating alerts. The relations among alerts can be used to create the attack graph.

The method introduced by Lee requires prior knowledge. In fact, they obtain the attack scenario from the correlations between alerts. They skip the phase of alert classification and use the methods introduced by others instead.

It can be said that building an attack scenario from classified alerts requires a completely network vision. The knowledge gained by the experts can be useful for creating an attack graph. In fact, the stronger the knowledge, the better the scenario will be. Hence, we are going to address the problem using data mining techniques and we are going to focus on the classification and correlation of alerts.

The proposed method includes the following steps:

a) The following steps are taken for each received alert.

b) Alerts are classified using the fuzzy method explained below. First, using a multi-layer neural network as the correlation engine the output for a pair of alerts is calculated.

If the desired output exceeds the predefined threshold, we skip into stage 3; otherwise a new hyper-alert is created, which will include the above alert.

c) The received alert is connected to all the available hyper-alerts. In addition, the square of the output from the correlation engine is used as the degree of membership for joining this alert to the existing alerts.

d) The attack patterns are compared to hyper-alerts in order to discover the attack mechanism. The fuzzy inference method can be used here as long as attack patterns are known.

2.4.1 Using a Multi-Layer Neural Network As The Correlation Engine

As it was explained in the Zhu-Ghorbani method, a multi-layer neural network can be used as a correlation engine. First, the neural network is examined using the following test samples. It shall be noted that the quality utilized in the Zhu-Ghorbani method is employed here as well.

After teaching the above network it is put into use as follows. First, the qualities obtained from the received alert are compared to the last alerts in the hyper-alerts. Next, the values are transferred to the correlation engine. The network output shows the probability of correlativity of two alerts. If the value exceeds the determined threshold (threshold=0.5), it is attached to the last alert in the hyper-alert. It shall be noted that using this method one alert can be appeared in several hyper-alerts. The output from the correlation engine is used as the degree of membership for joining one alert to the desired hyper-alert.

2.4.2 Using the Fuzzy Model for Establishing Connections Among Alerts

When the output from the correlation engine for each received alert and the last alert in the hyper-alert exceeds the threshold value, the alert is included in the hyper-alert. The output from the correlation engine is also used as the degree of membership for joining the alert to the hyper-alert.

After checking all alerts several hyper-alerts are obtained that can have one or several alerts (with different degrees of membership) in common. In order to illustrate the attack scenario the following steps are taken:

a) First prior knowledge obtained from experts should be turned into fuzzy rules.

b) Next using fuzzy inference model we relate hyper-alerts with attack scenarios.

It shall be noted that since the relationship of each alert with its succeeding alert is explained using degree of membership, the fuzzy inference model can be used.

The main problem is turning prior knowledge o fuzzy rules. The reason is that precision of these rules affects the demonstration of attack scenarios. In addition, turning prior knowledge into fuzzy rules is another challenge.

In the following the above mentioned method is compared to the Zhu-Ghorbani method.

First, the first 30 alerts in the DARPA 2000 data set are extracted using sniffing software. Next, a matrix is created based on the resulting alerts. The matrix includes 6 qualities including alert number, alert delivery time, alert source IP address, alert destination IP address, destination port number, and alert length. The described matrix is illustrated below.

The 30 alerts are later used as the input for the proposed algorithm as well as the algorithm introduced by Ghorbani and Zhu. The output from these algorithms is as follows:

a) Using the Ghorbani-Zhu method the total number of hyper-alerts was equal to 28. That is to say, there are only two hyper-alerts with two alerts and the rest of the hyper-alerts only include one alert. This quantitative output is far from reality because the combination of the above 30 alerts implies that a large number of alerts can be classified into one class.

b) Using the method proposed in this paper only 1 hyper-alert was obtained and all the other 30 alerts were interconnected.

2.4.3 Information on Alerts Exported From DARPA 2000 Dataset into Matrixes

Table 2.2: Information Exports From DARPA 2000 Dataset

Destination	Protocol	Length	Info
Falcon.eyrie.af.mil	TELNET	60	Telnet Data…
Delta.peach.mil	TCP	60	Telnet Data
Falcon.eyrie.af.mil	TELNET	60	63281>telnet [ACK] Seq=2 Ack=2 Win=33580 Len=0
Falcon.eyrie.af.mil	TCP	60	Telnet Data…
Delta.peach.mil	TCP	60	Telnet Data…
Falcon.eyrie.af.mil	TELNET	60	63281> telnet [ACK] Seq=4 Ack=4 Win=33580 Len=0
Falcon.eyrie.af.mil	TELNET	60	Telnet Data…
Delta.peach.mil	TELNET	60	Telnet Data…
Falcon.eyrie.af.mil	TCP	60	63281> telnet [ACK] Seq=5 Ack=5 Win=33580 Len=0
Falcon.eyrie.af.mil	TELNET	60	Telnet Data…
Delta.peach.mil	TELNET	60	Telnet Data…
Falcon.eyrie.af.mil	TCP	60	63281> telnet [ACK] Seq=6 Avk=6 Win=33580 Len=0
Falcon.eyrie.af.mil	TELNET	60	Telnet Data…
Delta.peach.mil	TELNET	60	Telnet Data…
Falcon.eyrie.af.mil	TCP	60	63281> telnet [ACK] Seq=7 Ack=7 Win=33580 Len=0
Falcon.eyrie.af.mil	TELNET	60	Telnet Data…

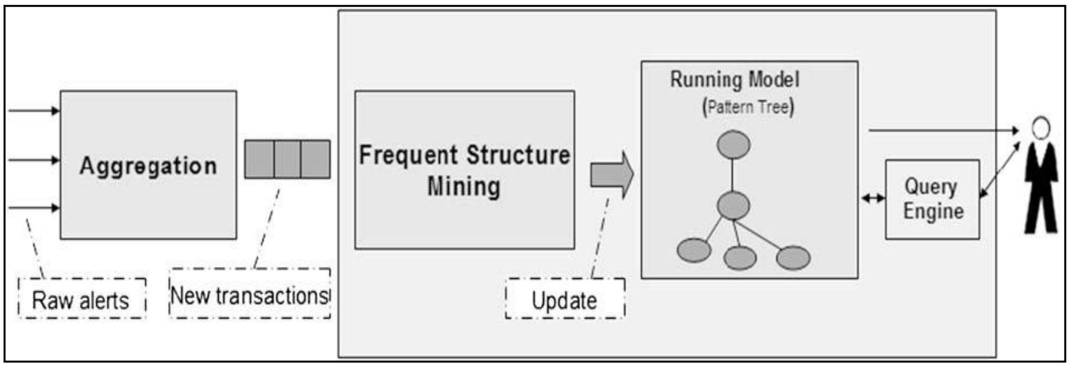
Delta.peach.mil	TELNET	60	Telnet Data…
Falcon.eyrie.af.mil	TCP	60	63281> telnet [ACK] Seq=8 Ack=8 Win=33580 Len=0
Falcon.eyrie.af.mil	TELNET	60	Telnet Data…
Delta.peach.mil	TELNET	60	Telnet Data…
Falcon.eyrie.af.mil	TCP	60	63281> telnet [ACK] Seq=9 Ack=9 Win=33580 Len=0
Falcon.eyrie.af.mil	TCP	60	Telnet Data…
Delta.peach.mil	TELNET	60	Telnet Data…
Falcon.eyrie.af.mil	TCP	60	63281> telnet [ACK] Seq=10 Ack=10 Win=33580 Len=0
Falcon.eyrie.af.mil	TELNET	60	Telnet Data…
Delta.peach.mil	TELNET	60	Telnet Data…

2.5 LITERATURE

In Zhu-Ghorbani method correlation probability between two alerts is calculated based on similarity of features of source IP, destination IP, destination port, type of alert, and timestamps using Multi-Layer Perception (MLP) neural network and Support Vector Machines (SVM). In this method, when a new alert is received, a hyper-alert that includes an alert with maximum correlation probability with the new alert is specified using MLP and SVM. If the detected correlation probability was less than correlation threshold, the new alert is not correlated with any alerts. If the calculated probability exceeded threshold, correlation probability of the new alert is calculated with all the available alerts in the detected hyper-alert. After that, the alert is correlated with the new alert whose difference of probability with the highest probability detected earlier is less than criterion of correlation sensitivity. If there is no alert for correlation, a new alert is placed in a new hyper-alert (Zhu and Ghorbani, 2006).

Following figure shows a framework presented for alert correlation. Unprocessed alerts are received continuously by integration unit. This unit correlates alerts to graph structures based on their connection information with respect to the source and destination of the alerts. Each structural pattern may show attack strategies or maybe the normal pattern created due to positive false alerts. The created patterns may change dynamically as long as they become fixed. The fixed structural patterns are transferred to the next unit to create a set of transactions for the following processes.

Figure 2.4: A framework for correlation between alerts



Source: R. Sadoddin and A. A. Ghorbani, (2009) *An incremental frequent structure mining framework for real-time alert correlation*. April 2009.

In this method, features of source IPs, destination IPs, attack classes, and timestamps are used for different alerts. Feature of port is not used in this method as frequent patterns are shown by data graph structures, which are nodes of network hosts and edges of the alerts issued between hosts. On the other hand, a port is not an unreliable feature source (as each intruder can easily change his/her port) and value of destination port in most attacks is not important (Sadoddin and Ghorbani, 2009).

In the method presented for creating candidate frequent patterns, transactions are created based on the connection information of corresponding alerts. Here, one method is presented for exploring frequent patterns incrementally and maintaining them in the reduced data structure (FP-tree).

FP-Growth algorithm was used for exploring sequential structures. FP-Growth algorithm uses FP_Tree, which is a compressed data structure for storing frequent candidate patterns a concept called 'source' was used in *D. Xu and P. Ning, "Privacy-preserving alert correlation: A concept hierarchy based approach"*, to show prerequisite and consequence of an attack. A 'source' can be a port, a service, etc. Prerequisite of an attack, input source, and its consequences is called 'output source'. In this method, the causal relationships between resources were prepared in the form of rules and they are used to create correlations between alerts. Two alerts are correlated when the output source of either of them include one of the input sources of the other and/or lead to them.

Minor compliance was used in this article. That is, if the result of an alert meets at least one of the prerequisites of another alert (regarding time relationship), those alerts will be correlated.

2.6 PROPOSED METHOD SUMMARY

Correlation of alerts has several steps as follows. First, alerts are classified after preprocessing. Then an attack scenario is created using the available alerts in a group. An attack scenario is strongly dependent on the earlier knowledge and classification quality. Earlier knowledge is meant the knowledge collected from professionals that can help to create an attack graph (that expresses attack scenario). The richer and more accurate the knowledge is, the presented scenarios will be better. Therefore, we intend to focus on a part to be able to solve the problem using data mining techniques. As a result, we will concentrate on how to classify and correlate alerts.

Our proposed method encompasses the following steps:

i. For all the received alerts, we do the following steps.

ii. Classification of alerts using the fuzzy method explained below.

First, we calculate output for a pair of alerts using MLP neural network as a correlation engine. We teach the above neural network using training samples.

If the relevant output were bigger than the predefined threshold, we would go through step 3; otherwise, we create a new hyper-alert and put above alert in it.

We connect the received alert to all the available hyper-alerts and we use the second output power of the correlation engine as membership degree of the alert to the present hyper-alerts.

2.6.1 Using Neural Network as a Correlation Engine

As explained in the method of Zhu and Ghorbani, a multi-layer neural network can be used as a correlation engine. First, we teach the neural network using the following training samples. The features we used here are

1- Source IP address, 2- Destination IP address, 3- Destination port number, and 4- To examine if destination IP address of the earlier alert is identical with the source IP address of the current alert

After teaching the above network, it is used as follows. Here, we compare the features extracted from the received alerts and the ending alerts in infra-alerts and give their values to the correlation engine. The network output shows correlation probability of the two alerts. If this value exceeded the predefined threshold (We assumed threshold value equal to 0.5.), we connect it to the ending alert in the above hyper-alert. In this method, one alert may appear in several hyper-alerts. We use output of correlation engine as membership degree of an alert to the relevant hyper-alert.

2.6.2 Using Fuzzy Classification to Establish Relationship between Alerts

When output of correlation engine exceeds threshold value for the received alert and final alert in a hyper-alert, we put the alert in that hyper-alert and use output of the correlation engine as membership degree of that alert to the hyper-alert.

After examining all alerts, we will have several hyper-alerts that may have common alerts (but with different membership degree).

2.7 HOW TO RUN THE PROGRAM

The program folder contains some files and subfolders. The first subfolder named stack contains stack class. Therefore we use correlationAl2 algorithm in making attack graph. The next subfolder named data contains data sets extracted from DARPA2000 data set using surveillance tools.

The alert correlating algorithm and making Force and victim hyper-alerts are present in correlationAl function. First, type the

```
dataIDS = xlsread('data\dataIDS2.xlsx');
```

Instruction in the command line till the data is read and is put in dataIDS variable. Then run the program like the following calling correlationAl function.

```
Correlational (dataIDS);
```

After finishing, the program returns a cell structure with the number of cells expressing the number of hyper-alerts, and each cell representing a hyper-alert. Each cell’s structure is like a matrix where the rows indicate alerts and the columns express their relationship. For instance, see the matrix below.

Table 2.3: Cell structure returns

1	0	0.0
2	1	0.9275
3	1	0.7550

Alert 1 is entered as the head of hyper-alerts which has no relations with any other alerts in the above matrix. Alert 2 is in relation with alert 1 (column 2, row 2 expresses this) and the amount of their relationship equals 0.9275 column 3, row 2).

The CorrCal function is called in line 20 of this program. It takes 2 alerts, calculates their correlation amount using the correlation engine, and returns the result.

The correlation engine is created in learnAl function. This function first calls the learning data set in line 9 and then creates a multi-layer neural network with following parameters in lines 10-18.

a) Number of input layers: 3
b) Number of first-layer neurons: 4
c) Number of mid-layer neurons: 4
d) Number of output neurons: 1

The designed network starts learning using the learning data sets in line 24. The learned network is finally saved in line 26 of the program. You can either execute the program to see how, or use the saved network.

The attack scenario is created (based on Zhu-Ghorbani method) in correlationAl2 function. The ACM matrix is loaded in line 12. It is the same force and victim matrix whose creation method is described in calculateACM function.

The difference between our method with the force and victim method is in the way of forming hyper-alerts and the way of alerts' correlation. Our function is presented in 'fuzzyModel' algorithm. We pay attention to properties extraction method for calculating correlation probability between the two models before analyzing the above function. Properties extraction is presented in featureMatching function. The data sets are loaded in lines 3-5 here. The first and second properties' values which equals the same source and target IP addresses respectively are calculated in lines 6-42. The next two properties which equal target port number and the same IP address for the present alert's source and the IP address for the previous alert's target are calculated in lines 44-56.

Parameters such as correlation threshold is set in fuzzyModel function lines 1-8 (note that relative names are used for variables and functions to understand the program).

A loop is placed in line 9 to examine all alerts. Then it is examined that the above alert can correlate with which final present alert in hyper-alerts in loop 17. The correlation probability calculated in line 22 is compared with the threshold. If greater, it is put in that hyper-alert and the correlation probability is used as membership degree. This is done in lines 22-26. Otherwise, then a new hyper-alert is created and the above alert is placed in it. This is done in lines 34-36. Finally, the cell containing all hyper-alerts is returned. Each cell represents one hyper-alert. To elaborate more, suppose we have called the above function as the following.

```
f = fuzzyModel (dataIDS);
```

Then f is a cell structure whose number represents hyper-alerts at the end of program run. Suppose f is a cell structure of 1*5. Then, this represents 5 hyper-alerts. Just type the instruction below in the command line to see the internal alerts of a hyper-alert.

```
F1 = f {1, 3}
```

Therefore F will contain alerts forming hyper-alert 3 (to show, type it without ;).

3. FINDINGS

It can be proved that this method leads to a better categorization. To do so, we assume that we received alert α_1. Probability of correlation of this alert with the two alerts, which are within two different hyper-alerts, close to one another and it exceeds the threshold (0.5) we defined for instance, probability of 0.6 for its correlation with the alert in the first hyper-alert and 0.65 for its correlation with the alert in the second hyper-alert. As noticed, such difference is negligible. According to other classification methods, assume that we put this alert in the first hyper-alert, while, in fact, it is related to the second hyper-alert. It is due to the fact that correlation engine is unable to show their correlation favorably. This might be due to the accuracy of a learning machine (Learning machine's accuracy cannot be hundred percent.) and/or due to lack of appropriate training. Therefore, by losing this alert in the second hyper-alert, we may not be able to extract attack scenario. (Assume a condition in which such mode is repeated several times.)

Table 3.1: Data For Learn

1	1	0	1	1	1	1
1	1	0	0	0	0	0.75
1	1	0	0	0.5	0.5	0.85
0.5	1	0	0	0.5	0.5	0.8
0.5	0.5	0	0	0.1	0.3	0
0	1	0	0	0.1	0.2	0
1	0.5	0	1	0.5	0.3	0.65
0	0	0	0	0	0	0
0.5	1	0	0	1	1	0.85
0.5	0.5	0	1	1	1	0.8
1	1	0	1	0	0	0.9
0.5	0.5	0	0	0.5	0	0
0	0	0	1	1	1	0.65
0	0	1	1	1	1	0.9
0	0	1	0	0.5	0	0.8
0	0	1	1	0.5	0.5	0.85
0	0	1	0	0	0	0.8
0.5	0.5	0	0	0.5	1	0

Now, assuming that we can have this alert in both hyper-alerts, we will be able to compensate defect of attack scenario by having the pertinent alert. We can consider constructing an attack scenario in a way to ignore construction algorithm of their scenario as soon as we observe the irrelevant alerts. It means that placing an alert in such hyper-alert cannot lead to confusion about attack scenario.

We tested our algorithm on 30 sample alerts out of all the alerts of *DARPA 2000* dataset and the result was as follows:

Using neural network and predefined threshold in Zhu-Ghorbani method, the alerts were classified into several groups. While we placed them in a group using their own method, this result was acceptable because all the alerts were somehow related to each other. Results can be shown on appendix A.12 and A.13.

Table 3.2: CorrelationAl Algorithm Output “correlationAl (dataIDS);”:

1	0	0
2	1	0.773381779
3	0	0
4	0	0
5	0	0
6	0	0
7	0	0
8	0	0
9	0	0
10	0	0
11	1	0.807486488
12	0	0
13	0	0
14	0	0
15	0	0
16	0	0
17	0	0
18	0	0
19	0	0
20	0	0
21	0	0
22	0	0
23	0	0
24	0	0
25	0	0
26	0	0
27	0	0
28	0	0
29	0	0
30	0	0

Table 3.3: FuzzyModel Algorithm Output “f = fuzzyModel (dataIDS);”:

1	0	0
2	1	0.773382
3	2	0.773382
4	3	0.999973
5	4	0.773382
6	5	0.773382
7	6	0.999973
8	7	0.773382
9	8	0.773382
10	9	0.999879
11	10	0.807486
12	11	0.807486
13	12	0.999879
14	13	0.773382
15	14	0.773382
16	15	0.999879
17	16	0.807486
18	17	0.807486
19	18	0.999879
20	19	0.773382
21	20	0.773382
22	21	0.999879
23	22	0.807486
24	23	0.807486
25	24	0.999973
26	25	0.807486
27	26	0.807486
28	27	0.999879
29	28	0.773382
30	29	0.773382

4. CONCLUSION

Here, we aimed to present a better method for correlating alerts. In our method, first, we use MLP as a correlation engine. This engine specifies probability of correlation of two alerts. Then we classified alerts using an algorithm and present them in the form of a hyper-alert. The advantage of this method is that one alert can be placed in several hyper-alerts simultaneously. If one alert is placed in another group by mistake, such advantage will not lead to non-extraction of attack scenario of a hyper-alert.

REFERENCES

Books

F. Cuppens and R. Ortalo (2000). *Lambda: A language to model a database for detection of attacks* .In Recent Advances in Intrusion Detection, vol. 1907 of LNCS, pp. 197–216, Springer Berlin Heidelberg.

Witten IH, Frank E. (2005). *Data Mining: practical machine learning tools and techniques.* San Francisco: Morgan Kaufmann Publishers.

Periodicals

B. Zhu, A. Ghorbani, 2006. Alert correlation for extracting attack strategies. *International Journal of Network Security*, vol. **3**, no. 3, pp. 244–258.

C. Granger, 1999 .Investigating causal relations by econometric models and cross spectral methods. *Econometrical*, vol. **34**, pp. 424–428.

J.J. Davis, J.C. Andrew, 2011. Data preprocessing for anomaly based network intrusion detection: a review. *Journal of Computers and Security*, vol. **30**, 353–375.

L. Wang, A. Liu, and S. Jajodia, 2006. Using attack graphs for correlating, hypothesizing, and predicting intrusion alerts. *Computer communications*, vol. **29**, no. 15, pp. 2917–2933.

O. M. Dain, R. K. Cunningham, 2002. Building scenarios from a heterogeneous alert stream. *In Proceedings of IEEE Workshop on Information Assurance and Security*, vol. **6**, (United States Military Academy, West Point, NY), pp. 231–235.

S.T. Eckmann, G. Vigna, and R. A. Kemmere, 2002.Statl: An attack language for state-based intrusion detection. *Journal of Computer Security*, vol. **10**, no. 1-2, pp. 71–103.

Other Publications

B. Morin, H. Debar, 2003. Correlation of intrusion symptoms: an application of chronicles. *In: Proc. Of the 6th Int. Conf. on Recent Advances in Intrusion Detection* (RAID'03), 2003, pp. 94–112.

C. Michel, L. Mé. , 2001. Adele: An attack description language for knowledge-based intrusion detection. *In Proceedings of the 16th Annual Working Conference on Information Security: Trusted Information: The New Decade Challenge*, vol. 193, pp. 353–368, Kluwer, B.V. Deventer, Netherlands.

D. Xu and P. Ning, 2005. Privacy-preserving alert correlation: A concept hierarchy based approach. *In Proceedings of the 21st Annual Computer Security Applications Conference (ACSAC),* pp. 537–546.

H. Ren, N. Stakhanova and A. Ghorbani, 2010 .An online adaptive approach to alert correlation *.In Proceedings of the 7th international conference on Detection of intrusions and malware, and vulnerability assessment, DIMVA'10,* pp. 153–172.

J. Chen, D. J. DeWitt, F. Tian and Y. Wang, 2000 .A scalable continuous query system for internet databases *.In Proceedings of ACM SIGMOD*, pp.379–390.

P. Ning, Y. Cui, and D. S. Reeves, 2002. Constructing attack scenarios through correlation of intrusion alerts. *In Proceedings of the 9th ACM conference on Computer and communications security, (Washington, DC, USA),* pp. 245–254

P. Ning, D. Xu, Christopher, G. Healey, and R. S. Amant, 2004 .Building attack scenarios through integration of complementary alert correlation method *.In Proceedings of the 11th Annual Network and Distributed System Security Symposium (NDSS),* pp. 97–111.

R. Agrawal, R. Srikant, 2000. Mining sequential patterns. *In Proceedings of the 11th IEEE International Conference on Data Engineering (ICDE), (Taipei, Taiwan),* pp. 3–14, IEEE Computer Society.

R. Sadoddin, A. A. Ghorbani, 2009 .An incremental frequent structure mining framework for real-time alert correlation . *Computers & Security*, 200905/06

Reza Sadoddin, 2006. Alert correlation survey, *Proceedings of the 2006 International Conference on Privacy Security and Trust Bridge the Gap between PST Technologies and Business Services* - PST 06 PST 06.

Seyed Hossein Ahmadinejad, 2009. Alert Correlation Using Correlation Probability Estimation and Time Windows. *In: International Conference on Computer Technology and Development*, 11/2009.

S. Saeed, Gabriel Maciá-Fernández, and Jesús E. Díaz-Verdejo, 2013.A model-based survey of alert correlation techniques .*Computer Networks*.

V. Holus, T. Parsons and P. O'Sullivan, J. Murphy, 2009 .Run-time correlation engine for system monitoring and testing. *In: Proc. of the 6th IEEE Int. Conf. on Autonomic Computing (ICAC-INDST '09)*, pp. 43–44.

X. Qin, W .lee. , 2003. Statistical causality analysis of InfoSec alert data. *In Proceedings of the 6th International Symposium on Recent Advances in Intrusion Detection (RAID), (Pittsburgh, PA)*, pp. 73–93.

APPENDICES

APPENDIX A.1: correlationAl FUNCTION

```
function hyperAlertList = correlationAl(ListOfAlert)

    corrThreshold = 0.5;
    corSensity = 0.1;
    hyperAlert = zeros(1,3);
    hyperAlertList = cell(1,1);
    idxHyperAlert = 0;
    preAlert = 0;
    for m=1:size(ListOfAlert,1)
        alert = m;
        maxCorr = 0;
        if(idxHyperAlert==0)
            idxHyperAlert = idxHyperAlert+1;
            hyperAlert(1,1) = alert;
            hyperAlertList{1,idxHyperAlert} = hyperAlert;
        else
            for n=1:size(hyperAlertList,2)
                hyperAlertS = hyperAlertList{1,n};
                for k=1:size(hyperAlertS,1)
                    probCorr =
corrCal(ListOfAlert(hyperAlertS(k,1),:),ListOfAlert(alert,:),preAlert);
                    if(probCorr>maxCorr)
                        maxCorr = probCorr;
                        maxIdxHyperA = n;
                        maxIdxA = k;
                    end
                end
            end
                if(maxCorr>corrThreshold)
                    hyperAlertSelected = hyperAlertList{1,maxIdxHyperA};
                    flagFind = 0;
                    for i=1:size(hyperAlertSelected,1)
                        if(i==maxIdxA)
                            continue;
                        end
                        probCorr =
corrCal(ListOfAlert(hyperAlertSelected(i,1),:),ListOfAlert(alert,:),preAlert);
                        if((maxCorr - probCorr)<corSensity)
                            l = size(hyperAlertSelected,1);
                            hyperAlertSelected(l+1,1) = alert;
                            hyperAlertSelected(l+1,2) = i;
                            hyperAlertSelected(l+1,3) = probCorr;
                            hyperAlertList{1,maxIdxHyperA} = hyperAlertSelected;
                            flagFind = 1;
                        end
                    end
                if(i==1)
                    l = size(hyperAlertSelected,1);
                    hyperAlertSelected(l+1,1) = alert;
                    hyperAlertSelected(l+1,2) = i;
                    hyperAlertSelected(l+1,3) = probCorr;
                    hyperAlertList{1,maxIdxHyperA} = hyperAlertSelected;
                    flagFind = 1;
                end
                if(flagFind==0)
                    hyperAlert(1,1) = alert;
                    idxHyperAlert = idxHyperAlert+1;
                    hyperAlertList{1,idxHyperAlert} = hyperAlert;
                end
            end
        end
        preAlert = ListOfAlert(m,:);
    end
end
```

APPENDIX A.2: correlationAl2 FUNCTION

```
function correlationAl2(listOfAlert)

    stackObj = stack;
    threshold = 0.1;
    r = randi(size(listOfAlert,1));
    stackObj = stackObj.inqueue(listOfAlert(r,1));
    graphAttack = listOfAlert(r,1);
    idxGraphAttack = 1;
    visitedGraph = zeros(size(listOfAlert,1),1);

    %acm = calculateACM(listOfAlert);
    load acm;

    isEmpty = stackObj.top;
    while(isEmpty>0)
        stackObj = stackObj.dequeue();
        alert =  stackObj.dequeuedElm;
        for i=1:size(acmMatrix,2)
            forwardCorrStr = acmMatrix(alert,i)/sum(acmMatrix(alert,:));
            if(forwardCorrStr>threshold)
                if(visited(i,1)==0)
                    stackObj = stackObj.inqueue(listOfAlert(i,1));
                    visitedGraph(i,1) = 1;
                end
                graphAttack(idxGraphAttack,2) = i;
                graphAttack(idxGraphAttack,3) = acmMatrix(alert,i);
                idxGraphAttack = idxGraphAttack+1;
            end
        end
        isEmpty = stackObj.top;
    end
end
```

APPENDIX A.3: calculateACM FUNCTION

```
function acmMatrix = calculateACM(listOfAlert)

    %%Load correlation Engine
        load ('netMlp.mat');
    %%End of Load correlation Engine

    preAlert = zeros(1,size(listOfAlert,2));
    for i=1:size(listOfAlert)
        alert = listOfAlert(i,:);
        for j=1:size(listOfAlert)
            [f1,f2,f3,f4] = featureMatching(listOfAlert(j,:),alert,preAlert);
            preAlert = listOfAlert(j,:);
            %% Load the Correlation Engine and Calculate probability of correlation
                f = [f1;f2;f3;f4;];
                corrProb = sim(net,f);
            %%End of Load the Correlation Engine and Calculate probability of
correlation

            acmMatrix(i,j) = corrProb;
        end
    end
end
```

APPENDIX A.4: readData FUNCTION

```
function readData()
    [data,path] = uigetfile('m2.csv');
    data = dataset('xlsfile',sprintf('%s\%s', path,data));
end
```

APPENDIX A.5: fuzzyModel FUNCTION

```
function hyperAlertList = fuzzyModel(ListOfAlert)

    corrThreshold = 0.5;
    %corSensity = 0.1;
    hyperAlert = zeros(1,3);
    hyperAlertList = cell(1,1);
    idxHyperAlert = 0;
    preAlert = 0;
    for m=1:size(ListOfAlert,1)
        alert = m;
        %maxCorr = 0;
        if(idxHyperAlert==0)
            idxHyperAlert = idxHyperAlert+1;
            hyperAlert(1,1) = alert;
            hyperAlertList{1,idxHyperAlert} = hyperAlert;
        else
            for n=1:size(hyperAlertList,2)
                hyperAlertS = hyperAlertList{1,n};
                %for k=1:size(hyperAlertS,1)
                l = size(hyperAlertS,1);
                probCorr =
corrCal(ListOfAlert(hyperAlertS(l,1),:),ListOfAlert(alert,:),preAlert);
                if(probCorr>corrThreshold)
                    hyperAlertS(l+1,1) = alert;
                    hyperAlertS(l+1,2) =  hyperAlertS(l,1);
                    hyperAlertS(l+1,3) = probCorr;
                    hyperAlertList{1,n} = hyperAlertS;
%                     for i=1:size(hyperAlertS,1)-1
%                         probCorr =
corrCal(ListOfAlert(hyperAlertS(i,1),:),ListOfAlert(alert,:),preAlert);
%                         hyperAlertS(l+1,i+1,1) = i;
%                         hyperAlertS(l+1,i+1,2) = probCorr^2;
%                         hyperAlertList{1,n} = hyperAlertS;
%                     end
                else
                    hyperAlert(1,1) = alert;
                    idxHyperAlert = idxHyperAlert+1;
                    hyperAlertList{1,idxHyperAlert} = hyperAlert;
                end
            end
        end
        preAlert = ListOfAlert(m,:);
    end
end
```

APPENDIX A.6: featureMatching FUNCTION

```
function [f1,f2,f3,f4] = featureMatching(hyperAlertS,alert,preAlert)

    addressIP = xlsread('data\addressIP.xlsx');
    hyAlIP = zeros(1,8);
    alertIP = zeros(1,8);
   %% Calculation of f1,f2
    for k=3:4
        for i=1:size(addressIP,1)
            if(hyperAlertS(1,k)==addressIP(i,1))
                for j=2:5
                    hyAlIP(j-1,:) = bitget(addressIP(i,j),8:-1:1,'uint8');
                end
            end

            if(alert(1,k)==addressIP(i,1))
                for j=2:5
                    alertIP(j-1,:) = bitget(addressIP(i,j),8:-1:1,'uint8');
                end
            end
        end

        %alIP = num2str(alertIP(1,5));
        %hyAIP = num2str(hyAlIP(1,5));
        for i=1:8
            match = 0;
            for j=i:8
                if(alertIP(4,j)==hyAlIP(4,j))
                    match = match+1;
                else
                    break;
                end
            end
            matchT(1,i) = match;
        end
        matchT = sort(matchT,'descend');
        if(k==3)
            f1 = (24+matchT(1,1))/32;
        else
            f2 = (24+matchT(1,1))/32;
        end
    end
   %%End of Calculation of f1,f2

  %% Calculate another features
    if(hyperAlertS(1,5)==alert(1,5))
        f3 = 1;
    else
        f3 = 0;
    end

    if(preAlert(1,4)==alert(1,3))
        f4 = 1;
    else
        f4 = 0;
    end
  %%End of Calculate another features
end
```

APPENDIX A.7: featureMatchForCls FUNCTION

```
function [f1,f2] = featureMatchForCls(hyperAlertS,alert)

    addressIP = xlsread('data\addressIP.xlsx');
    hyAlIP = zeros(1,8);
    alertIP = zeros(1,8);
   %% Calculation of f1,f2
    for k=3:4
        for i=1:size(addressIP,1)
            if(hyperAlertS(1,k)==addressIP(i,1))
                for j=2:5
                    hyAlIP(j-1,:) = bitget(addressIP(i,j),8:-1:1,'uint8');
                end
            end

            if(alert(1,k)==addressIP(i,1))
                for j=2:5
                    alertIP(j-1,:) = bitget(addressIP(i,j),8:-1:1,'uint8');
                end
            end
        end

        %alIP = num2str(alertIP(1,5));
        %hyAIP = num2str(hyAlIP(1,5));
        for i=1:8
            match = 0;
            for j=i:8
                if(alertIP(4,j)==hyAlIP(4,j))
                    match = match+1;
                else
                    break;
                end
            end
            matchT(1,i) = match;
        end
        matchT = sort(matchT,'descend');
        if(k==3)
            f1 = (24+matchT(1,1))/32;
        else
            f2 = (24+matchT(1,1))/32;
        end
    end
   %%End of Calculation of f1,f2
```

APPENDIX A.8: learnAl FUNCTION

```
%This function learn a neural network to produce a probability of
%correlation between two alerts.
%Notice that the p and t parameters must be this way: p is a matrix which
%it's rows show the features and it's columns show the elements. t also
% is a matrix which it's rows show the class(Label)s and its columns show
% elements.
function learnAl()

    load('dataNet.mat');
    MinAndMax = zeros(4,1);
    MinAndMax = [MinAndMax ones(4,1)];
    net = newff(MinAndMax,[4,1],{'tansig','tansig'});
    init(net);

    net.trainParam.show = 50;
    net.trainParam.lr = 0.05;
    net.trainParam.epochs = 300;
    net.trainParam.goal = 1e-5;

    p = dataForLearn(:,1:4);
    p = reshape(p,4,18);
    t = dataForLearn(:,7);
    t = reshape(t,1,18);
    net = train(net,p,t);

    save netMlp net;
end
```

APPENDIX A.9: STACK FUNCTION

```
classdef stack
   properties
      table = zeros(1,1);
      top = 0;
      dequeuedElm = 0;
   end

   methods
      function obj = inqueue(obj,value)
         obj.top = (obj.top)+1;
         t = obj.top;
         obj.table(t,1) = value; %('farshid');
      end

      function obj = dequeue(obj)
         t = obj.top;
         obj.dequeuedElm = obj.table(t,1);
         obj.table(t) = [];
         obj.top = (obj.top)-1;
      end
   end
end
```

APPENDIX A.10: corrCal FUNCTION

```
function corrProb = corrCal(hyperAlertS,alert,preAlert)

    addressIP = xlsread('data\addressIP.xlsx');
    hyAlIP = zeros(1,8);
    alertIP = zeros(1,8);
   %% Calculation of f1,f2
    for k=3:4
        for i=1:size(addressIP,1)
            if(hyperAlertS(1,k)==addressIP(i,1))
                for j=2:5
                    hyAlIP(j-1,:) = bitget(addressIP(i,j),8:-1:1,'uint8');
                end
            end

            if(alert(1,k)==addressIP(i,1))
                for j=2:5
                    alertIP(j-1,:) = bitget(addressIP(i,j),8:-1:1,'uint8');
                end
            end
        end

        %alIP = num2str(alertIP(1,5));
        %hyAIP = num2str(hyAlIP(1,5));
        for i=1:8
            match = 0;
            for j=i:8
                if(alertIP(4,j)==hyAlIP(4,j))
                    match = match+1;
                else
                    break;
                end
            end
            matchT(1,i) = match;
        end
        matchT = sort(matchT,'descend');
        if(k==3)
            f1 = (24+matchT(1,1))/32;
        else
            f2 = (24+matchT(1,1))/32;
        end
    end
   %%End of Calculation of f1,f2

  %% Calculate another features
    if(hyperAlertS(1,5)==alert(1,5))
        f3 = 1;
    else
```

```
        f3 = 0;
    end

    if(preAlert(1,4)==alert(1,3))
        f4 = 1;
    else
        f4 = 0;
    end
  %%End of Calculate another features

    %% Load the Correlation Engine and Calculate probability of correlation
        load ('netMlp.mat');
        f = [f1;f2;f3;f4;];
        corrProb = sim(net,f);
     %%End of Load the Correlation Engine and Calculate probability of correlation
End
```

APPENDIX A.11: NET MLP

val =

Neural Network

dimensions:

connections:

subobjects:

functions:

weight and bias values:

methods:

APPENDIX A.12: ACM Matrix Output

0.9978	0.06643	0.95568	0.9978	0.06643	0.95568	0.9978	0.06643	0.95568	0.99221
0.75924	0.95568	0.06643	0.75924	0.95568	0.06643	0.75924	0.95568	0.06643	0.82715
0.9978	0.06643	0.95568	0.9978	0.06643	0.95568	0.9978	0.06643	0.95568	0.99221
0.9978	0.06643	0.95568	0.9978	0.06643	0.95568	0.9978	0.06643	0.95568	0.99221
0.75924	0.95568	0.06643	0.75924	0.95568	0.06643	0.75924	0.95568	0.06643	0.82715
0.9978	0.06643	0.95568	0.9978	0.06643	0.95568	0.9978	0.06643	0.95568	0.99221
0.9978	0.06643	0.95568	0.9978	0.06643	0.95568	0.9978	0.06643	0.95568	0.99221
0.75924	0.95568	0.06643	0.75924	0.95568	0.06643	0.75924	0.95568	0.06643	0.82715
0.9978	0.06643	0.95568	0.9978	0.06643	0.95568	0.9978	0.06643	0.95568	0.99221
0.99221	0.24931	0.99221	0.99221	0.24931	0.99221	0.99221	0.24931	0.99221	0.9978
0.1497	0.02459	0.1497	0.1497	0.02459	0.1497	0.1497	0.02459	0.1497	-0.0006
0.99221	0.24931	0.99221	0.99221	0.24931	0.99221	0.99221	0.24931	0.99221	0.9978
0.9978	0.06643	0.95568	0.9978	0.06643	0.95568	0.9978	0.06643	0.95568	0.99221
0.75924	0.95568	0.06643	0.75924	0.95568	0.06643	0.75924	0.95568	0.06643	0.82715
0.9978	0.06643	0.95568	0.9978	0.06643	0.95568	0.9978	0.06643	0.95568	0.99221
0.99221	0.24931	0.99221	0.99221	0.24931	0.99221	0.99221	0.24931	0.99221	0.9978
0.1497	0.02459	0.1497	0.1497	0.02459	0.1497	0.1497	0.02459	0.1497	-0.0006
0.99221	0.24931	0.99221	0.99221	0.24931	0.99221	0.99221	0.24931	0.99221	0.9978
0.9978	0.06643	0.95568	0.9978	0.06643	0.95568	0.9978	0.06643	0.95568	0.99221
0.75924	0.95568	0.06643	0.75924	0.95568	0.06643	0.75924	0.95568	0.06643	0.82715
0.9978	0.06643	0.95568	0.9978	0.06643	0.95568	0.9978	0.06643	0.95568	0.99221
0.99221	0.24931	0.99221	0.99221	0.24931	0.99221	0.99221	0.24931	0.99221	0.9978

0.1497	0.02459	0.1497	0.1497	0.02459	0.1497	0.1497	0.02459	0.1497	-0.0006
0.99221	0.24931	0.99221	0.99221	0.24931	0.99221	0.99221	0.24931	0.99221	0.9978
0.99221	0.24931	0.99221	0.99221	0.24931	0.99221	0.99221	0.24931	0.99221	0.9978
0.1497	0.02459	0.1497	0.1497	0.02459	0.1497	0.1497	0.02459	0.1497	-0.0006
0.99221	0.24931	0.99221	0.99221	0.24931	0.99221	0.99221	0.24931	0.99221	0.9978
0.9978	0.06643	0.95568	0.9978	0.06643	0.95568	0.9978	0.06643	0.95568	0.99221
0.75924	0.95568	0.06643	0.75924	0.95568	0.06643	0.75924	0.95568	0.06643	0.82715
0.9978	0.06643	0.95568	0.9978	0.06643	0.95568	0.9978	0.06643	0.95568	0.99221

0.1497	0.99221	0.9978	0.06643	0.95568	0.99221	0.1497	0.99221	0.9978	0.06643
0.02459	0.24931	0.06643	0.95568	0.06643	0.82715	0.02459	0.24931	0.06643	0.95568
0.1497	0.99221	0.9978	0.06643	0.95568	0.99221	0.1497	0.99221	0.9978	0.06643
0.1497	0.99221	0.9978	0.06643	0.95568	0.99221	0.1497	0.99221	0.9978	0.06643
0.02459	0.24931	0.06643	0.95568	0.06643	0.82715	0.02459	0.24931	0.06643	0.95568
0.1497	0.99221	0.9978	0.06643	0.95568	0.99221	0.1497	0.99221	0.9978	0.06643
0.1497	0.99221	0.9978	0.06643	0.95568	0.99221	0.1497	0.99221	0.9978	0.06643
0.02459	0.24931	0.06643	0.95568	0.06643	0.82715	0.02459	0.24931	0.06643	0.95568
0.1497	0.99221	0.9978	0.06643	0.95568	0.99221	0.1497	0.99221	0.9978	0.06643
-0.0006	0.95568	0.99221	0.24931	0.99221	0.9978	-0.0006	0.95568	0.99221	0.24931
0.95568	-0.0006	0.69072	0.02459	0.1497	-0.0006	0.95568	-0.0006	0.69072	0.02459
-0.0006	0.95568	0.99221	0.24931	0.99221	0.9978	-0.0006	0.95568	0.99221	0.24931
0.1497	0.99221	0.9978	0.06643	0.95568	0.99221	0.1497	0.99221	0.9978	0.06643

0.02459	0.24931	0.06643	0.95568	0.06643	0.82715	0.02459	0.24931	0.06643	0.95568
0.1497	0.99221	0.9978	0.06643	0.95568	0.99221	0.1497	0.99221	0.9978	0.06643
-0.0006	0.95568	0.99221	0.24931	0.99221	0.9978	-0.0006	0.95568	0.99221	0.24931
0.95568	-0.0006	0.69072	0.02459	0.1497	-0.0006	0.95568	-0.0006	0.69072	0.02459
-0.0006	0.95568	0.99221	0.24931	0.99221	0.9978	-0.0006	0.95568	0.99221	0.24931
0.1497	0.99221	0.9978	0.06643	0.95568	0.99221	0.1497	0.99221	0.9978	0.06643
0.02459	0.24931	0.06643	0.95568	0.06643	0.82715	0.02459	0.24931	0.06643	0.95568
0.1497	0.99221	0.9978	0.06643	0.95568	0.99221	0.1497	0.99221	0.9978	0.06643
-0.0006	0.95568	0.99221	0.24931	0.99221	0.9978	-0.0006	0.95568	0.99221	0.24931
0.95568	-0.0006	0.69072	0.02459	0.1497	-0.0006	0.95568	-0.0006	0.69072	0.02459
-0.0006	0.95568	0.99221	0.24931	0.99221	0.9978	-0.0006	0.95568	0.99221	0.24931
-0.0006	0.95568	0.99221	0.24931	0.99221	0.9978	-0.0006	0.95568	0.99221	0.24931
0.95568	-0.0006	0.69072	0.02459	0.1497	-0.0006	0.95568	-0.0006	0.69072	0.02459
-0.0006	0.95568	0.99221	0.24931	0.99221	0.9978	-0.0006	0.95568	0.99221	0.24931
0.1497	0.99221	0.9978	0.06643	0.95568	0.99221	0.1497	0.99221	0.9978	0.06643
0.02459	0.24931	0.06643	0.95568	0.06643	0.82715	0.02459	0.24931	0.06643	0.95568
0.1497	0.99221	0.9978	0.06643	0.95568	0.99221	0.1497	0.99221	0.9978	0.06643

0.95568	0.99221	0.1497	0.99221	0.99221	0.1497	0.99221	0.9978	0.06643	0.95568
0.06643	0.82715	0.02459	0.24931	0.24931	0.02459	0.24931	0.06643	0.95568	0.06643

0.95568	0.99221	0.1497	0.99221	0.99221	0.1497	0.99221	0.9978	0.06643	0.95568
0.95568	0.99221	0.1497	0.99221	0.99221	0.1497	0.99221	0.9978	0.06643	0.95568
0.06643	0.82715	0.02459	0.24931	0.24931	0.02459	0.24931	0.06643	0.95568	0.06643
0.95568	0.99221	0.1497	0.99221	0.99221	0.1497	0.99221	0.9978	0.06643	0.95568
0.95568	0.99221	0.1497	0.99221	0.99221	0.1497	0.99221	0.9978	0.06643	0.95568
0.06643	0.82715	0.02459	0.24931	0.24931	0.02459	0.24931	0.06643	0.95568	0.06643
0.95568	0.99221	0.1497	0.99221	0.99221	0.1497	0.99221	0.9978	0.06643	0.95568
0.99221	0.9978	-0.0006	0.95568	0.9978	-0.0006	0.95568	0.99221	0.24931	0.99221
0.1497	-0.0006	0.95568	-0.0006	0.74015	0.95568	-0.0006	0.69072	0.02459	0.1497
0.99221	0.9978	-0.0006	0.95568	0.9978	-0.0006	0.95568	0.99221	0.24931	0.99221
0.95568	0.99221	0.1497	0.99221	0.99221	0.1497	0.99221	0.9978	0.06643	0.95568
0.06643	0.82715	0.02459	0.24931	0.24931	0.02459	0.24931	0.06643	0.95568	0.06643
0.95568	0.99221	0.1497	0.99221	0.99221	0.1497	0.99221	0.9978	0.06643	0.95568
0.99221	0.9978	-0.0006	0.95568	0.9978	-0.0006	0.95568	0.99221	0.24931	0.99221
0.1497	-0.0006	0.95568	-0.0006	0.74015	0.95568	-0.0006	0.69072	0.02459	0.1497
0.99221	0.9978	-0.0006	0.95568	0.9978	-0.0006	0.95568	0.99221	0.24931	0.99221
0.95568	0.99221	0.1497	0.99221	0.99221	0.1497	0.99221	0.9978	0.06643	0.95568
0.06643	0.82715	0.02459	0.24931	0.24931	0.02459	0.24931	0.06643	0.95568	0.06643
0.95568	0.99221	0.1497	0.99221	0.99221	0.1497	0.99221	0.9978	0.06643	0.95568
0.99221	0.9978	-0.0006	0.95568	0.9978	-0.0006	0.95568	0.99221	0.24931	0.99221
0.149	-	0.955	-	0.740	0.955	-	0.690	0.024	0.149

7	0.0006	68	0.0006	15	68	0.0006	72	59	7
0.99221	0.9978	-0.0006	0.95568	0.9978	-0.0006	0.95568	0.99221	0.24931	0.99221
0.99221	0.9978	-0.0006	0.95568	0.9978	-0.0006	0.95568	0.99221	0.24931	0.99221
0.1497	-0.0006	0.95568	-0.0006	0.74015	0.95568	-0.0006	0.69072	0.02459	0.1497
0.99221	0.9978	-0.0006	0.95568	0.9978	-0.0006	0.95568	0.99221	0.24931	0.99221
0.95568	0.99221	0.1497	0.99221	0.99221	0.1497	0.99221	0.9978	0.06643	0.95568
0.06643	0.82715	0.02459	0.24931	0.24931	0.02459	0.24931	0.06643	0.95568	0.06643
0.95568	0.99221	0.1497	0.99221	0.99221	0.1497	0.99221	0.9978	0.06643	0.95568

APPENDIX A.13: ACM Calculation Output “acm = calculateACM (dataIDS);”

0.99997	-0.6575	0.99986	0.99997	-0.6575	0.99986	0.99997	-0.6575	0.99986	0.99988
0.77338	0.99986	-0.6575	0.77338	0.99986	-0.6575	0.77338	0.99986	-0.6575	0.50983
0.99997	-0.6575	0.99986	0.99997	-0.6575	0.99986	0.99997	-0.6575	0.99986	0.99988
0.99997	-0.6575	0.99986	0.99997	-0.6575	0.99986	0.99997	-0.6575	0.99986	0.99988
0.77338	0.99986	-0.6575	0.77338	0.99986	-0.6575	0.77338	0.99986	-0.6575	0.50983
0.99997	-0.6575	0.99986	0.99997	-0.6575	0.99986	0.99997	-0.6575	0.99986	0.99988
0.99997	-0.6575	0.99986	0.99997	-0.6575	0.99986	0.99997	-0.6575	0.99986	0.99988
0.77338	0.99986	-0.6575	0.77338	0.99986	-0.6575	0.77338	0.99986	-0.6575	0.50983
0.99997	-0.6575	0.99986	0.99997	-0.6575	0.99986	0.99997	-0.6575	0.99986	0.99988
0.99988	-0.348	0.99988	0.99988	-0.348	0.99988	0.99988	-0.348	0.99988	0.99997
-0.7326	-0.6882	-0.7326	-0.7326	-0.6882	-0.7326	-0.7326	-0.6882	-0.7326	-0.6763
0.99988	-0.348	0.99988	0.99988	-0.348	0.99988	0.99988	-0.348	0.99988	0.99997
0.99997	-0.6575	0.99986	0.99997	-0.6575	0.99986	0.99997	-0.6575	0.99986	0.99988
0.77338	0.99986	-0.6575	0.77338	0.99986	-0.6575	0.77338	0.99986	-0.6575	0.50983
0.99997	-0.6575	0.99986	0.99997	-0.6575	0.99986	0.99997	-0.6575	0.99986	0.99988
0.99988	-0.348	0.99988	0.99988	-0.348	0.99988	0.99988	-0.348	0.99988	0.99997
-0.732	-0.688	-0.732	-0.732	-0.688	-0.732	-0.732	-0.688	-0.732	-0.676

6	2	6	6	2	6	6	2	6	3
0.99988	-0.348	0.99988	0.99988	-0.348	0.99988	0.99988	-0.348	0.99988	0.99997
0.99997	-0.6575	0.99986	0.99997	-0.6575	0.99986	0.99997	-0.6575	0.99986	0.99988
0.77338	0.99986	-0.6575	0.77338	0.99986	-0.6575	0.77338	0.99986	-0.6575	0.50983
0.99997	-0.6575	0.99986	0.99997	-0.6575	0.99986	0.99997	-0.6575	0.99986	0.99988
0.99988	-0.348	0.99988	0.99988	-0.348	0.99988	0.99988	-0.348	0.99988	0.99997
-0.7326	-0.6882	-0.7326	-0.7326	-0.6882	-0.7326	-0.7326	-0.6882	-0.7326	-0.6763
0.99988	-0.348	0.99988	0.99988	-0.348	0.99988	0.99988	-0.348	0.99988	0.99997
0.99988	-0.348	0.99988	0.99988	-0.348	0.99988	0.99988	-0.348	0.99988	0.99997
-0.7326	-0.6882	-0.7326	-0.7326	-0.6882	-0.7326	-0.7326	-0.6882	-0.7326	-0.6763
0.99988	-0.348	0.99988	0.99988	-0.348	0.99988	0.99988	-0.348	0.99988	0.99997
0.99997	-0.6575	0.99986	0.99997	-0.6575	0.99986	0.99997	-0.6575	0.99986	0.99988
0.77338	0.99986	-0.6575	0.77338	0.99986	-0.6575	0.77338	0.99986	-0.6575	0.50983
0.99997	-0.6575	0.99986	0.99997	-0.6575	0.99986	0.99997	-0.6575	0.99986	0.99988

-0.7326	0.99988	0.99997	-0.6575	0.99986	0.99988	-0.7326	0.99988	0.99997	-0.6575
-0.6882	-0.348	-0.6575	0.99986	-0.6575	0.50983	-0.6882	-0.348	-0.6575	0.99986
-0.7326	0.99988	0.99997	-0.6575	0.99986	0.99988	-0.7326	0.99988	0.99997	-0.6575
-0.7326	0.99988	0.99997	-0.6575	0.99986	0.99988	-0.7326	0.99988	0.99997	-0.6575

-0.6882	-0.348	-0.6575	0.99986	-0.6575	0.50983	-0.6882	-0.348	-0.6575	0.99986
-0.7326	0.99988	0.99997	-0.6575	0.99986	0.99988	-0.7326	0.99988	0.99997	-0.6575
-0.7326	0.99988	0.99997	-0.6575	0.99986	0.99988	-0.7326	0.99988	0.99997	-0.6575
-0.6882	-0.348	-0.6575	0.99986	-0.6575	0.50983	-0.6882	-0.348	-0.6575	0.99986
-0.7326	0.99988	0.99997	-0.6575	0.99986	0.99988	-0.7326	0.99988	0.99997	-0.6575
-0.6763	0.99986	0.99988	-0.348	0.99988	0.99997	-0.6763	0.99986	0.99988	-0.348
0.99986	-0.6763	0.77107	-0.6882	-0.7326	-0.6763	0.99986	-0.6763	0.77107	-0.6882
-0.6763	0.99986	0.99988	-0.348	0.99988	0.99997	-0.6763	0.99986	0.99988	-0.348
-0.7326	0.99988	0.99997	-0.6575	0.99986	0.99988	-0.7326	0.99988	0.99997	-0.6575
-0.6882	-0.348	-0.6575	0.99986	-0.6575	0.50983	-0.6882	-0.348	-0.6575	0.99986
-0.7326	0.99988	0.99997	-0.6575	0.99986	0.99988	-0.7326	0.99988	0.99997	-0.6575
-0.6763	0.99986	0.99988	-0.348	0.99988	0.99997	-0.6763	0.99986	0.99988	-0.348
0.99986	-0.6763	0.77107	-0.6882	-0.7326	-0.6763	0.99986	-0.6763	0.77107	-0.6882
-0.6763	0.99986	0.99988	-0.348	0.99988	0.99997	-0.6763	0.99986	0.99988	-0.348
-0.7326	0.99988	0.99997	-0.6575	0.99986	0.99988	-0.7326	0.99988	0.99997	-0.6575
-0.6882	-0.348	-0.6575	0.99986	-0.6575	0.50983	-0.6882	-0.348	-0.6575	0.99986

-0.7326	0.99988	0.99997	-0.6575	0.99986	0.99988	-0.7326	0.99988	0.99997	-0.6575
-0.6763	0.99986	0.99988	-0.348	0.99988	0.99997	-0.6763	0.99986	0.99988	-0.348
0.99986	-0.6763	0.77107	-0.6882	-0.7326	-0.6763	0.99986	-0.6763	0.77107	-0.6882
-0.6763	0.99986	0.99988	-0.348	0.99988	0.99997	-0.6763	0.99986	0.99988	-0.348
-0.6763	0.99986	0.99988	-0.348	0.99988	0.99997	-0.6763	0.99986	0.99988	-0.348
0.99986	-0.6763	0.77107	-0.6882	-0.7326	-0.6763	0.99986	-0.6763	0.77107	-0.6882
-0.6763	0.99986	0.99988	-0.348	0.99988	0.99997	-0.6763	0.99986	0.99988	-0.348
-0.7326	0.99988	0.99997	-0.6575	0.99986	0.99988	-0.7326	0.99988	0.99997	-0.6575
-0.6882	-0.348	-0.6575	0.99986	-0.6575	0.50983	-0.6882	-0.348	-0.6575	0.99986
-0.7326	0.99988	0.99997	-0.6575	0.99986	0.99988	-0.7326	0.99988	0.99997	-0.6575

0.99986	0.99988	-0.7326	0.99988	0.99988	-0.7326	0.99988	0.99997	-0.6575	0.99986
-0.6575	0.50983	-0.6882	-0.348	-0.348	-0.6882	-0.348	-0.6575	0.99986	-0.6575
0.99986	0.99988	-0.7326	0.99988	0.99988	-0.7326	0.99988	0.99997	-0.6575	0.99986
0.99986	0.99988	-0.7326	0.99988	0.99988	-0.7326	0.99988	0.99997	-0.6575	0.99986
-0.6575	0.50983	-0.6882	-0.348	-0.348	-0.6882	-0.348	-0.6575	0.99986	-0.6575
0.99986	0.99988	-0.732	0.99988	0.99988	-0.732	0.99988	0.99997	-0.657	0.99986

		6			6			5	
0.99986	0.99988	-0.7326	0.99988	0.99988	-0.7326	0.99988	0.99997	-0.6575	0.99986
-0.6575	0.50983	-0.6882	-0.348	-0.348	-0.6882	-0.348	-0.6575	0.99986	-0.6575
0.99986	0.99988	-0.7326	0.99988	0.99988	-0.7326	0.99988	0.99997	-0.6575	0.99986
0.99988	0.99997	-0.6763	0.99986	0.99997	-0.6763	0.99986	0.99988	-0.348	0.99988
-0.7326	-0.6763	0.99986	-0.6763	0.80749	0.99986	-0.6763	0.77107	-0.6882	-0.7326
0.99988	0.99997	-0.6763	0.99986	0.99997	-0.6763	0.99986	0.99988	-0.348	0.99988
0.99986	0.99988	-0.7326	0.99988	0.99988	-0.7326	0.99988	0.99997	-0.6575	0.99986
-0.6575	0.50983	-0.6882	-0.348	-0.348	-0.6882	-0.348	-0.6575	0.99986	-0.6575
0.99986	0.99988	-0.7326	0.99988	0.99988	-0.7326	0.99988	0.99997	-0.6575	0.99986
0.99988	0.99997	-0.6763	0.99986	0.99997	-0.6763	0.99986	0.99988	-0.348	0.99988
-0.7326	-0.6763	0.99986	-0.6763	0.80749	0.99986	-0.6763	0.77107	-0.6882	-0.7326
0.99988	0.99997	-0.6763	0.99986	0.99997	-0.6763	0.99986	0.99988	-0.348	0.99988
0.99986	0.99988	-0.7326	0.99988	0.99988	-0.7326	0.99988	0.99997	-0.6575	0.99986
-0.6575	0.50983	-0.6882	-0.348	-0.348	-0.6882	-0.348	-0.6575	0.99986	-0.6575
0.99986	0.99988	-0.7326	0.99988	0.99988	-0.7326	0.99988	0.99997	-0.6575	0.99986
0.99988	0.99997	-0.676	0.99986	0.99997	-0.676	0.99986	0.99988	-0.348	0.99988

		3			3				
-0.7326	-0.6763	0.99986	-0.6763	0.80749	0.99986	-0.6763	0.77107	-0.6882	-0.7326
0.99988	0.99997	-0.6763	0.99986	0.99997	-0.6763	0.99986	0.99988	-0.348	0.99988
0.99988	0.99997	-0.6763	0.99986	0.99997	-0.6763	0.99986	0.99988	-0.348	0.99988
-0.7326	-0.6763	0.99986	-0.6763	0.80749	0.99986	-0.6763	0.77107	-0.6882	-0.7326
0.99988	0.99997	-0.6763	0.99986	0.99997	-0.6763	0.99986	0.99988	-0.348	0.99988
0.99986	0.99988	-0.7326	0.99988	0.99988	-0.7326	0.99988	0.99997	-0.6575	0.99986
-0.6575	0.50983	-0.6882	-0.348	-0.348	-0.6882	-0.348	-0.6575	0.99986	-0.6575
0.99986	0.99988	-0.7326	0.99988	0.99988	-0.7326	0.99988	0.99997	-0.6575	0.99986