Design of a Network-Based Anomaly Detection System Using VFDT Algorithm

Naseer Alwan Hussein

Submitted to the
Institute of Graduate Studies and Research
in partial fulfilment of the requirements for the Degree of

Master of Science
in
Computer Engineering

Eastern Mediterranean University
May 2014
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Elvan Yılmaz
Director

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

Prof. Dr. IşıkAybay
Chair, Department of Computer Engineering

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

Assoc. Prof. Dr. Mohammed Salamah
Supervisor

Examining Committee

1. Prof. Dr. Rza Bashirov

2. Assoc. Prof. Dr. Mohammed Salamah

3. Asst. Prof. Dr.Gurcu Oz

ABSTRACT

Despite the rapid progress in information technology, the problem of protecting computer and network security remained a major challenge for most researchers, especially after the expansion of networks and evolution of technology and the increasing number of network users and the internet. Networks need some tools for protection, such as firewall, Intrusion Detection Systems (IDSs) and Intrusion Prevention System (IPS). The aim of this thesis is to build a Network-based Anomaly Detection System (NADS). This system depends on the normal behavior of the network, in that it can distinguish each abnormal behavior. This system can work in two modes, online and offline modes. Very Fast Decision Tree (VFDT) algorithm was used to build the classifier for intrusions. VFDT is one of the data mining algorithms that deal with high data streams in a very short time. Experimental results demonstrated that NADS system is highly successful in detecting known and unknown attacks by 93%.

Keywords: Network Security, Intrusion Detection, Very Fast Decision Tree Algorithm, KKD CUP99 dataset.

ÖZ

Bilgi teknolojilerindeki hızlı gelişmelere rağmen bilgisayar ve ağ güvenliğinin sağlanması birçok araştırmacı için, özellikle ağ sayılarının çoğalması, teknolojinin değişmesi, sayıları artmakta olan ağ kullanıcıları ve İnternet gibi nedenlerle, çözülmesi zor bir konu olarak kalmıştır. Var olan ağların korunma için güvenlik duvarı, güvenlik ihlali tespit ve önleme sistemleri gibi çeşitli araçlara ihtiyacı bulunmaktadır. Bu tez çalışmasının amacı ağ merkezli bir anormal saldırı tespit sistemi geliştirmektir (NADS). Adı geçen sistems ağın normal davranışına bağlı olarak anormal davranış biçimini tespit etme görevini sürdürüp çevirimiçi ve çevirimdışı olmak üzere iki biçimde çalışmaktadır. Ağ ihlallerinin sınıflandırılması için Çok Hızlı Karar Verme Ağacı (VFDT) algoritması kullanılmıştır. Çok Hızlı Karar Verme Ağacı kısa sürede gerçekleşen yüksek veri akışı ile ilgilenen veri madenciliği algoritmalarından biridir. Deneysel sonuçlar,ağ, ağ merkezli anormal saldırı tespit sisteminin bilinen (NADS) ve bilinmeyen saldırıları tespit etmekte % 93 başarılı olduğunu göstermiştir.

Anahtar Kelimeler: Ağ Güvenliği, İhlal Tespiti,Çok Hızlı Karar Verme Ağacı Algoritması, KKD CUP99 veri grubu.

DEDICATION

I want to seize the opportunity to fully thank my parents; my father and my mother and my brothers, who were always beside me spiritually, supporting me with all their truth feelings, praying for me achieve success and exist my and their dream by getting my master diploma from this reputed university.

For each and every one whom once was a barrier rock in my life path, for those who made my journey of life complicated, for whom once abused me directly and indirectly, to all those failure, weakness and loss moments, to my dreams that I have always dreamt of achieving and to those times when I touched frustration loneliness and misery.

Thank you from the bottom of my heart, without you I wouldn’t have achieved success.

ACKNOWLEDGMENT

No words can describe my appreciation to my supervisor, Dr. Muhammed Salamah.

I want to take the opportunity as well to thank my parents whom without their inseparable support and prayers I wouldn't succeed. Firstly My Father, Alwan Hussein Ali, and the person for whom I should thank for planting fundaments of my knowledge, teaching me the joy of intellectual pursuit. Secondly I want to thank my dear Mother, Waheba Homad Muhammed, for she is the one who sincerely raised me with her caring and gentle love. Also I want to thank all my brothers.

Finally, I would like to thank everybody who was part of the success in my thesis, knowing that I sadly express my apology that I could not mention all with their personal names.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ADAM	Audit Data Analysis and Mining.
ANN	Artificial Neural Networks.
BN	Bayes Net.
CART	Classification and Regression Trees.
CM	Confusion Matrix.
CPT	Cost per Test.
DARPA	Defence Advanced Research Project Agency.
DB	Data Base.
DoS	Denial of Service.
DR	Detection Rate.
DT	Detection Tree.
FAR	False Alarm Rate.
FL	Fuzzy Logic.
FN	False Negative.
FP	False Positive.
IDDM	Intrusion Discovery Data Mining.
HB	Hoeffding Bound.
HIDS	Host-based Intrusion Detection System.
HT	Hoeffding Tree.
ICMP	Internet Control Message Protocol.
ID	Intrusion Detection.
ID3	Iterative Dichotomiser 3.

IDDM	Intrusion Detection Data Mining.
IDS	Intrusion Detection System.
IDSs	Intrusion Detection Systems.
IG	Information Gain.
IP	Internet Protocol.
IPS	Intrusion Prevention System.
ISP	Internet Service Provider.
ITI	Incremental Tree Induction.
KDD	Knowledge Discovery and Data.
KDD CUP99	Knowledge Discovery and Data Mining CUP1999.
LAN	Local Area Network.
MAC	Media Address Control.
MARS	Multivariate Adaptive Regression Spines.
MINDS	Minnesota Intrusion Detection System.
MLP	Multilayer Perception.
NADS	Network-based Anomaly Detection System.
NB	Naïve Bayesian.
NIDS	Network-based Intrusion System.
NN	Neural Network.
PART	Partial Audit Reclusion Tree.
PDU	Protocol Data Unit.
PHAD	Packet Header Anomaly Detection.
PSP	Percentage of Successful Prediction.

R2L	Remote to Local.
RF	Random Forest.
ROC	Receiver Operation Characteristic.
SLIQ	Supervised Learning In Ques.
SPRINT	Scalable Parallelizable Induction of Decision Tree.
SVM	Support Vector Machine.
SOM	Self-Organizating Map.
TCP	Transmission Control Protocol.
Winpcap	Windows Packet Capturing.
TN	True Negative.
TP	True Positive.
TPR	True Positive Rate.
U2R	User to Root.
UDP	User Datagram Protocol.
VFDT	Very Fast Decision Tree.
TCP/IP	Transmission Control Protocol/Internet Protocol.

Chapter 1

INTRODUCTION

1.1 Introduction

In our society, information is becoming increasingly dependent on rapid access and interactive processing. As this demand increases, more information is being stored on computers. The proliferation of inexpensive computers and computer networks has worsened the problem of unauthorized access and tampering with data. Increasing connectivity not only provides access to large and varied sources of data more quickly than ever before, it also provides an access path to data from virtually anywhere on the network. With an increased understanding of how systems work, intruders have become highly skilled at diagnosing weaknesses in systems, and exploiting them to obtain such privileges that they can do anything on the system. They also use patterns that are difficult to trace and identify. Computer systems are therefore not likely to remain safe in the nearest future because of the intruder's acts. Therefore, we must have measures in place to detect security breaches, i.e., identify intruders and their methods of intrusion [1].

Intrusion Detection System (IDS) is highly significant and recommended as the final defense in the overall protection scheme of a computer system. There are progresses not only to detect successful breaches of security, but also to monitor varying attempts to breach the system security, and also provides important information for timely counter measures [2].

1.2 Problem Statement

Monitoring the whole network traffic in a network is a challenging issue. Overlap of the protocols “protocol layering” makes it difficult to extract the features of packets quickly and monitor the network. Also, there is a difficulty in knowing whether a contact is normal or attack.

1.3 Motivation

Accuracy and efficiency are two very important performance measures for the IDS system. It is therefore necessary to provide valuable information for the IDS analyst who monitors the computer system and the network. The use of decision tree as a classification technique helps us to achieve that. In addition, it is important to work in real-time, or as close to real-time as possible. Real-time operation is necessary for the IDS analyst so as to help take counter measures against attacks before they do much harm to the protected systems.

1.4 Aim of the Thesis

The aim of this thesis is to build a network-based anomaly NADS system that helps to detect abnormal behavior of network traffic. This can be achieved by using fast data mining algorithm. This system is characterized with processing and analyzing of high-speed network traffic; to discover and accurately identify new attacks by reducing the False Alarm (FA) rate to the minimal; detecting the intrusion in real time and making use of known attacks pattern in the train phase to increase detection ratio.

1.5 Thesis Outlines

The layout of the remainder of the thesis and the contents of the individual chapters are reviewed briefly. The thesis is divided into five chapters. Chapter two presents theoretical background which contains a generic IDS model, Knowledge Discovery

Data mining (KDD CUP99 data set) description, enhanced decision tree algorithms and very fast decision tree algorithm. Chapter three shows the proposed system and system architecture. Chapter four shows the implementation and results from the proposed system and then compared the system with other classification algorithms. Chapter five finally presents conclusions and suggestions for future work.

Chapter 2

THEORETICAL REVIEW

2.1 Introduction

With the tremendous growth of network-based services and sensitive information on networks, network security is getting more and more importance than ever. Intrusion poses a serious security risk in a network environment. The ever growing new intrusion types have a serious problem for their detection. The human labeling of the available network audit data instances is usually tedious, time consuming and expensive.

2.2 Intrusion Detection System (IDS)

Intrusion is any set of actions that attempt to compromise the integrity, confidentiality, and availability of a computer resource. This definition disregards the success or failure of those actions, so it also corresponds to attacks against a computer system. The problem of identifying actions that attempt to compromise the integrity, confidentiality, or availability of a computer resource is called intrusion detection [3].

The purpose of an IDS product is to monitor the network system for any type of attacks. An attack might be signaled by something as simple as a program that could modify the user name or could be a complex attack that involves sequence of events spanning multiple systems. IDSs are classified through system monitors because they

usually depend on auditing information provided from the systems logs or data gathered by sniffing network traffic [4].

However, there are also Intrusion detection systems that do not operate in real time, either because of the nature of the analysis they perform or because they are geared for forensic analysis (analysis of what has happened in the past on a system). The definition of an Intrusion detection system does not include, preventing the intrusion from occurring, but only detecting it and reporting the operations. There are some Intrusion detection systems that try to react when they detect an unauthorized action. Such reaction usually includes stopping the damage, for example, by terminating a network connection [5].

IDS is therefore needed as another wall to protect computer systems. The central elements to Intrusion detection engine are: resources to be protected in a target system, i.e., user accounts file systems, system kernels…etc; models that characterize the "normal" or "legitimate" behavior of these resources; Techniques that compare the actual system activities with the established models, and identify those that are "abnormal" or "intrusive" [5].

The process of intrusion-analysis can be separated into four stages as below [6]:

1. *Preprocessing*: When data is collected from an IDS sensor, the data is organized for classification. The preprocessing will help us to determine the format the data is put into, which is usually some canonical format or could be a structured database.
2. *Analysis*: The analysis stage begins after the preprocessing stage is completed and it is applied to all the records in the database. The data record is

compared with the knowledge base, and the data record will either be logged as an intrusion event or it will be dropped.

3. *Responses*: When the data record has been logged as an intrusion, a response can be initiated. This response contains an alert and passively collection information about the intrusion.
4. *Refinement*: This stage is responsible for the correctness of the intrusion.

The authors in [7] introduced a generic model for intrusion detection that is shown in Figure 2.1. The model is widely applicable to intrusion detection systems today. The three main components are the event generator, the activity profile, and the rule set. Early intrusion detection systems often relied on expert system techniques and consequently the idea of a rule base appears in the generic model.

Event generator is the component that provides information about system energizing. Events are derived from system audit trails, from network traffic, or from application of specific subsystems such as firewalls or authentication servers. Rule set is best thought of as the interface that decides whether an intrusion has occurred or not. Rule-based expert systems frequently were the preferred inferences tool for early IDSs. The best way to think of the rule set is as a generic detector engine examining events and state data using models, rules, patterns, and statistics to flag intrusive behavior. Activity profile maintains the state of the system or network being monitored. As events appear from the data source, variables in the activity profile are updated. New variables might be created as well depending on the action stated by the rule set.

The presence of some event might trigger the rule base to learn and add a new rule. If the rule set detects a threshold change in the activity profile, one response could be to alter the types, frequency, or details of event emitted from the event generator [7].

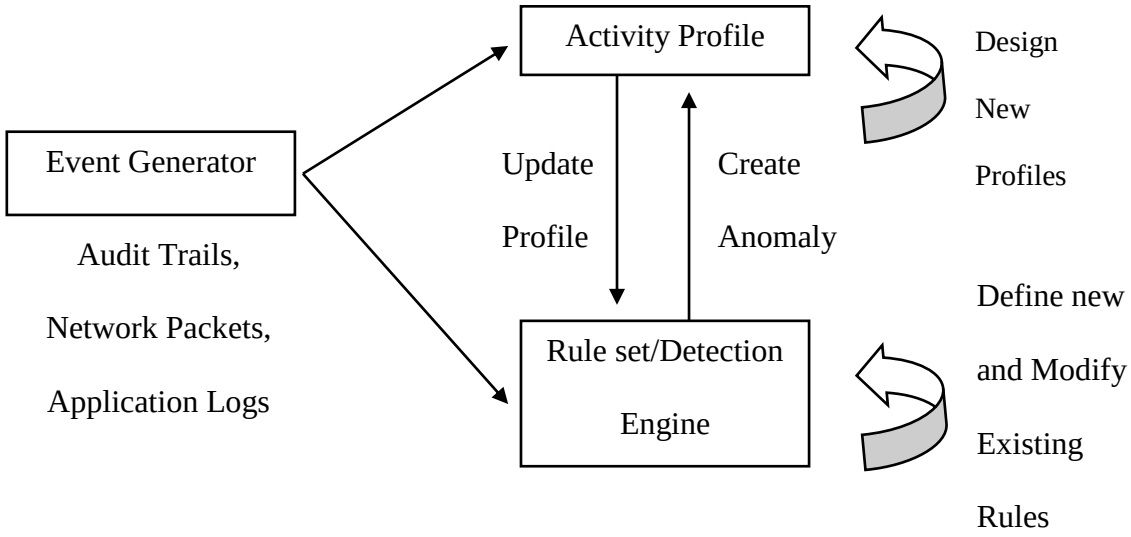

Figure 2.1: A Generic intrusion detection model [7].

Each of the three main subsystems could run on different nodes in a network. Intrusion detection system works on three main types of source information [8]:

1. *Network*: Data packets are collected and analyzed for signs of intrusion.

2. *Hosts*: Operating system and computer system details, such as memory and processor use, user activities, and applications running are examined for indications of misuse.

3. *Application*: Application logs "such as web server" can be examined for signs of attacks.

If the data is incomplete, detection abilities could be degraded. If the data is incorrect, the intrusion detection system could stop detecting certain intrusions and give its users a false sense of security [9].

2.2.1 Misuse Detection

Misuse detection identifies intrusions by matching monitored events to patterns or signatures of attacks. The attack signatures are the characteristics associated with successful known attacks. The major advantage of misuse detection is that the method possesses high accuracy in detecting known attacks. However, its detection ability is limited by the signature database. Unless new attacks are transformed into signatures and added to the database, a misuse-based IDS cannot detect any attack of this type. Different techniques such as expert systems, signature analysis, and state transition analysis are utilized in misuse detection [10].

2.2.2 Anomaly Detection

Anomaly detection assumes that intrusions are anomalies that necessarily differ from normal behavior. Basically, anomaly detection establishes a profile for normal operation and marks the activities that deviate significantly from the profile as attacks. The main advantage of anomaly detection is that it can detect unknown attacks. However, this advantage is paid for in terms of a high false positive rate because, in practice, anomalies are not necessarily intrusive. Moreover, anomaly detection cannot detect the attacks that do not obviously deviate from normal activities [10].

2.3 KDD CUP 99 Data set Description

Since 1999, Knowledge Discovery Data mining "KDD CUP99" has been the most wildly used data set for the evaluation of anomaly detection methods. This data set is prepared by Stolfo et al. It is built based on the data captured in DARPA'98 IDS evaluation program. DARPA'98 is about 4 gigabytes of compressed raw “binary” tcp dump data of 7 weeks of network traffic, which can be processed into about 5 million

connection records, each with about 100 bytes. The two weeks of test data have around 2 million connection records.

KDD training dataset consists of approximately 4,900,000 single connection vectors each of which contains 41 features and is labeled as either normal or an attack, with exactly one specific attack type [11]. The simulated attacks fall in one of the following four categories.

Denial of Service Attack (DoS): is an attack in which the attacker makes some computing or memory resource too busy or too full to handle legitimate requests, or denies legitimate users access to a machine.

User to Root Attack (U2R): is a class of exploit in which the attacker starts out with access to a normal user account on the system (perhaps gained by sniffing passwords, a dictionary attack, or social engineering) and is able to exploit some vulnerability to gain root access to the system.

Remote to Local Attack (R2L): occurs when an attacker who has the ability to send packets to a machine over a network but who does not have an account on that machine exploits some vulnerability to gain local access as a user of that machine.

Probing Attack (probe): is an attempt to gather information about a network of computers for the apparent purpose of circumventing its security controls.

Figure 2.2 shows the classification of the 41 features of the KDD CUP99 dataset sorted in a descending order through the information gain ratio. Most of the features

have Information Gain (IG) under the average of the datasets, “IG average = 0.22”. In fact, only 20 features are above IG average. This shows that the original dataset has data concentration in a small group of values. The features result in a convergence of connection categories within a small group of values which have little significant to describe a node behavior. This indicates that the original dataset may contain irrelevant data for the IDS and so needs to be optimized.

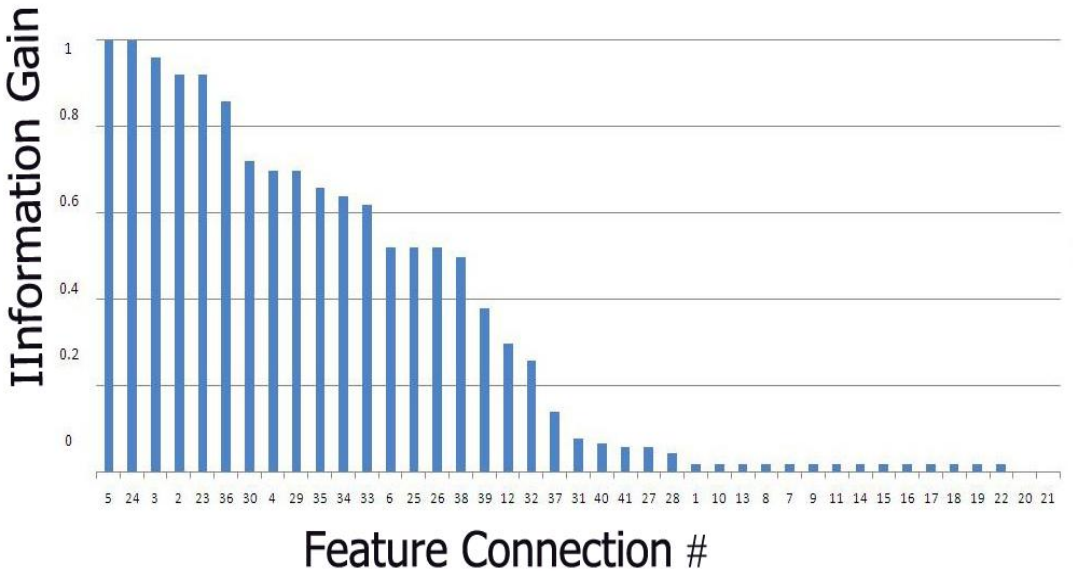

Figure 2.2: The maximum information gain for each feature [10].

2.4 Literature Review

This section surveys some of the recent and most related works to the proposed system.

The authors in [12] proposed the use of a data mining framework for building intrusion detection models. This framework consists of classification, association rules, and frequent episodes programs, which can be used to automatically construct detection models. They used the set of relevant system features to compute inductively learned, process raw audit data from the send mail system, call data and

the network TCP dump data and then summarized into connection records attributes. This approach applies on two general data mining algorithms: association rules algorithm, and the frequent episodes algorithm.

The authors in [13] proposed Audit Data Analysis and Mining (ADAM), a real-time anomaly detection system that uses data mining techniques to detect intrusions. ADAM uses a combination of association rules mining and classification to discover attacks in a TCP dump. ADAM uses a classifier which has been previously trained to classify the suspicious connections as a known type of attack, unknown type or a normal connection.

ADAM performs anomalies detection in two phases: the training phase, and the testing phase. In the training phase, it uses a data stream for which we know the types of the attack. The attack-free parts of the stream are fed into a module that performs off-line association rules discovery. The output of this module is a profile of rules that we call "normal". The profile along with the training data set is also fed into a module that uses a combination of a dynamic, on-line algorithm for association rules, whose output consists of frequent item sets that characterize attacks to the system. These item sets, along with a set of features extracted from the data stream by a feature selection module are used as the training set for a classifier "decision tree". This whole phase takes place off-line before using the system to detect intrusions. In the testing phase, the actual detection of intrusions is implemented.

The author in [14] proposed the combination of multiple host-based detectors using decision tree. This method uses conventional measures for intrusion detection and modeling methods appropriate to each measure. Statistical Rule-base method is used

to model these measures which are combined with decision tree. The proposed detection method has a good performance because it can model normal behaviors from various perspectives.

The decision tree used here is the C4.5 algorithm. The result shows that the combined detection method dramatically reduces the false-positive error rate against various types of intrusion.

The authors in [15] suggested a network-based intrusion detection system by using "Apriority Algorithm". The system gathered its information from the User Datagram Protocol (UDP) and Internet Control Message Protocol (ICMP), which work at the internet and transport layers within the TCP/IP communication model. This captured information is stored at a data warehouse. It found the relationships between attributes values that are stored at the data warehouse, and these relationships presents the normal behavior of the network and calculate the deviation from the normal behavior. It calculates the minimum distance from the normal behavior by entering different weights for the data warehouse, according to their importance within the network then testing the system on a local computer network which is built for this purpose.

The authors in [16] proposed the use of two algorithms: back propagation algorithm and C4.5 algorithm for intrusion detection. Since these algorithms are mainly applicable to misuse detection, it shows that the definition of anomaly detection not only takes into account normal profiles, but also handles known attacks and explores supervised machine learning algorithm; particularly neural networks and decision trees for intrusion detection. In fact, decision trees induction algorithm has proved its

efficiency in predicting the different classes of unlabeled data in Knowledge Discovery Databases (KDD CUP99). Test data set contains attacks such as Denial of Service (DoS), probe, User to Root (U2R) and Remote to Local (R2L). Experimental results demonstrate that while neural networks are highly successful in detecting known attacks, decision trees are more interesting to detect new attacks.

In [17], the authors proposed an approach which applies one of the efficient data mining algorithms called naïve Bayes for anomaly based network intrusion detection. Experimental results on the KDD CUP 99 data set showed the novelty of the approach in detecting network intrusion. It is observed that the technique performs better in terms of false positive rate, cost, and computational time when applied to KDD CUP 99 data set compared to a back propagation neural network based approach.

In [18], the authors designed a fuzzy logic-based system for effectively identifying the intrusion activists within a network. The fuzzy logic-based system helps to detect an intrusion behavior of the networks, since the rule base contains a better set of rules. The system uses mechanical method for generation of fuzzy rules that are obtained from the definite rules using frequent items. The experiments and evaluations of this intrusion detection system are performed with the KDD CUP 99 Intrusion detection dataset. The experiments results show that fuzzy logic-based system achieved higher precision in identifying whether the records are normal or attack.

The authors in [19], proposed a novel hybrid model for misuse and anomaly detection. The C4.5 based detection tree separates the network traffic into normal

and attack categories. The normal traffic is sent to anomaly detector and parallel attacks are sent to a decision trees based classifier for labeling with specific attack type. The model is trained and tested on two disjoint dataset provided in the KDD CUP 99. Results show the achievement of high detection rate for misuse and anomaly detection techniques. Summarizes the previous works that are related with the subject of our study in Table 2.1.

Table 2.1: Summary of different related works.

Seq.	Reference	Methodology	Algorithm	Online
1	[8]	Data mining approaches for intrusion detection and classification based anomaly detection. Features extraction from header, content and statistical packets.	Association Rules, Frequent Episodes	No
2	[14]	ADAM Audit Data Analysis and mining, association rules and classification based anomaly detection. Features extraction from header, content and statistical packets.	Association Rules, Decision tree	Yes
3	[15]	Combining multiple host-based detectors using decision tree. Use conventional measures for intrusion detection.	C4.5	No
4	[16]	Apriori algorithm in finding the association rules. Calculate the deviation from the normal behavior.	Apriori	No
5	[17]	Neural networks vs. decision trees for intrusion detection. Enhance the notion of anomaly detection and then used both neural networks and decision trees for intrusion detection.	Back propagation neural networks, C4.5	Yes
6	[18]	Apply one of the efficient data mining algorithms called Naïve Bayes for anomaly based network intrusion detection.	Naïve Bayes	No
7	[20]	Evaluate performance of a comprehensive set of classifier algorithms using KDD CUP 99 dataset.	Bayes Net, J48, Decision Table, Ripper,MLP	No
8	[21]	Attacks classification in adaptive intrusion detection using decision tree. Features extraction from header, content and statistical packets.	ID3,C4.5	No
9	[23]	Implementation of anomaly detection technique using machine learning algorithms. Features extraction from header, content and statistical packets.	K-Means, ID3	No
10	[24]	Designed fuzzy logic-based system for effectively identifying the intrusion activities within a network.	Fuzzy Logic	No
11	[25]	Proposed novel hybrid model for misuse and anomaly detection. Features extraction from header, content and statistical packets.	C4.5	No

2.5 Very Fast Decision Tree (VFDT) Algorithm

VFDT is a high-performance data mining system based on Hoeffding trees. Many classification learning methods have been proposed, of which the decision tree learning method is commonly used, because it is fast and the description of classifiers that it derives is easily understood. VFDT deals with data streams. As data arrives, this data stream grows gradually while the data is classified [20].

VFDT does not accumulate the examples in main memory, because it can gradually grow without waiting for the arrival of all the examples. The construction algorithm of the VFDT accumulates only the classes of examples and the synchronous occurrence frequency of attribute values in each node to decrease the consumption of memory and processing time. Hence, instead of accumulating examples in a decision tree the VFDT gradually grows as examples are received to create leaf nodes which grow into branches from only the root node. VFDT is different from classical Decision Tree (DT) algorithms. Classical DT receives all examples as input, and is called an offline type decision tree. Therefore, it cannot be applied to data streams. On the other hand, a VFDT construction in which new examples arrive in sequence at short intervals in a data stream and huge numbers of examples accumulate is called an online type decision tree [21]. This algorithm is presented in Figure 2.3.

Algorithm Anomaly Detection

Input: Sequence Vector of Features

Output: Class of Vectors

1. Select specific features from offline data set vectors and arrange them in file.
2. Read vector of features file(with class) to learn model in training phase
3. While. T. Do
4. If apply model in on line mode(test phase) Then
5. Capture network packets and extract basic features only as vectors from network traffic.
6. If vector is normal Than
7. Mark the vector is normal.
8. Else
9. Mark the vector is abnormal.
10. Else
11. Capture network packets and extract basic, content and statistical Feature from several sessions and save in file (without class).
12. Read features file (without class) to test connection.
13. If vector is normal Than
14. Mark the vector is normal.
15. Else
16. Mark the vector is abnormal.
17. If interrupt the system then
18. Exit
19. Else
20. End
21. Return class

Figure 2.3: The VFDT algorithm [21].

In this thesis, we apply one of the efficient data mining algorithms called Very Fast Decision Tree (VFDT) for anomaly based network intrusion detection. Experimental results on the Knowledge Discovery Data Mining (KDDCUP99) data set show that

our approach is effective in detecting network intrusion. It is observed that the proposed technique performs better in terms of false alarm rate, cost, and computational time when applied to KDD99 data sets compared with other algorithms [22].

Chapter 3

THE PROPOSED NADS SYSTEM

3.1 Introduction

This chapter presents the proposed NADS system by using Very Fast Decision Tree (VFDT) algorithm. It also classifies and identifies each connection record to normal or intrusion. Finally, the results are displayed as alarm report. The main function of IDS is data collection, pre-processing, classification and response. Data collection is a stage where data is captured from network traffic and then used to train and test the system. Pre-processing stage works on configuring the data which helps to build an effective classification system. VFDT classification algorithm is used to build the proposed NADS system and classify the network behavior in offline and online modes. The alerting model is the last stage of this proposed system, here; it displays the important information to security analyst in order to analyze the risks, and then take appropriate actions.

3.2 System Architecture

The system architecture is shown in Figure 3.1. It presents a generic architecture of an incremental Network-based Anomaly Detection System (NADS).

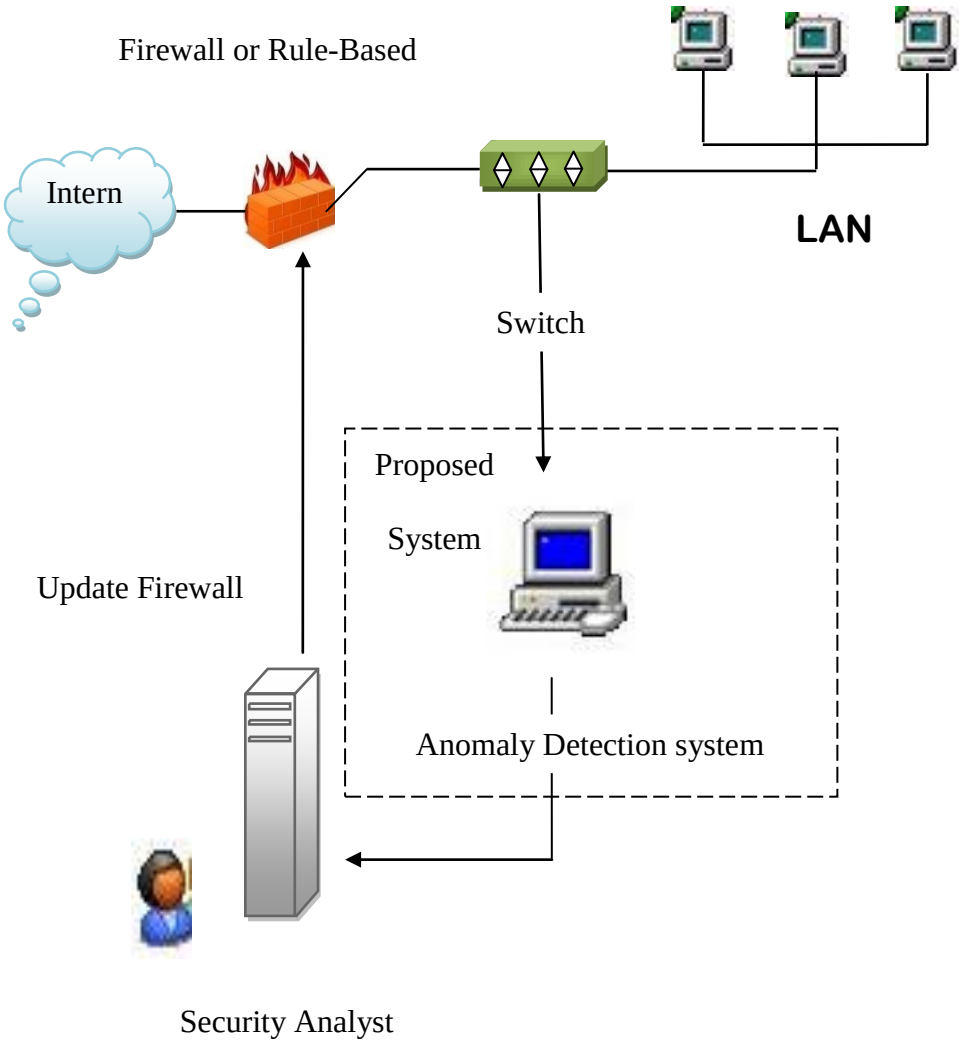

Figure 3.1: A Generic architecture of an NADS

The most important part is the anomaly detection system. It detects abnormal behavior, and then saves the alerts in a computer as records, and a copy of this alert record is sent to the network administrator "Security Analyst". Thereafter, appropriate actions are taken, by updating the database of the protection systems on the network, e.g. HIDS…etc.

3.3 The Proposed System

The system consists of two phases: the training phase and the testing phase. The training phase is shown in Figure 3.2. We have three steps and they are summarized as follows:

- KDD CUP99 dataset, it is used in the training phase.
- The operation mode, it contains the training phase.
- The proposed NADS system uses the VFDT algorithm, and it classifies the connections as normal or abnormal.

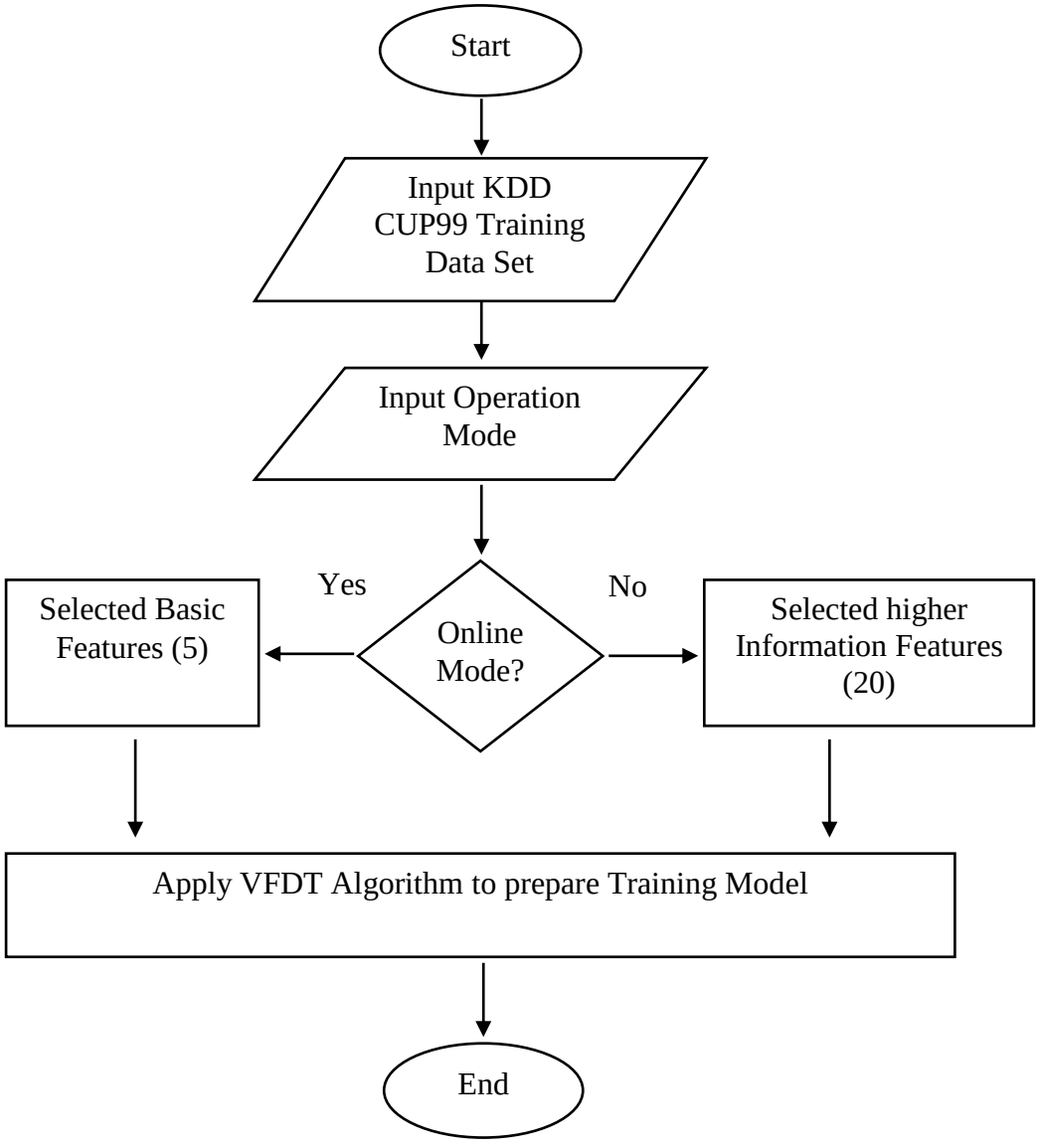

Figure 3.2: Block diagram of the training phase of the proposed system.

The testing phase is shown in Figure 3.3. We have five steps are summarized as follows:

- The packet capture is the first step in the testing phase. This part is concerned with capturing the packets which passed through the whole network.
- Pre-processing refers to the process of extracting information about packet connection from header and data in addition to construction of new statistical features.
- The operation mode, it contains the testing phase.
- The proposed NADS system uses the VFDT algorithm, and it classifies the connections as normal or abnormal.
- The alert report is the last step in the proposed NADS system.

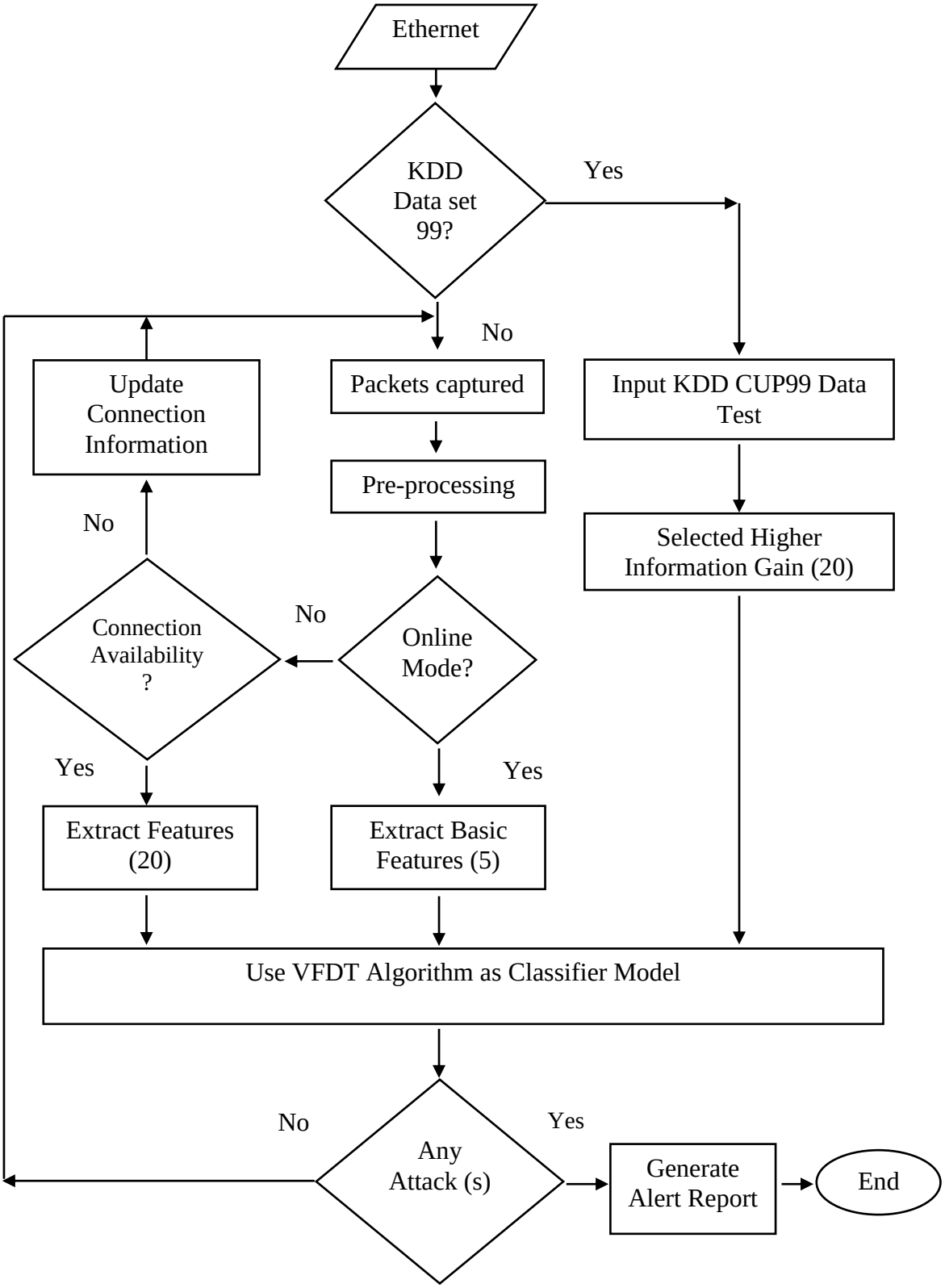

Figure 3.3: Block diagram of the testing phase of the proposed system.

This system can work in two modes: online and offline. This depends on the type of features used to train and test the system. The online mode works on the basic features which are extracted from the header of packets only, such as (protocol type,

service, flag, src_bytes, and dst_bytes). Then it organizes these features as connection records to enter into the model directly so as to determine the nature of the connection as soon as possible, to give the appropriate response.

The offline mode works on the basic features, and statistical features. These features are extracted from the header and the content of the packets. The statistical features are calculated when each session is terminated, and the features are organized as connection record. A large number of packets are needed to be calculated. These connections are corrected, and then stored in the file in the form of connection records in order to be entered into the classified model to test their classes. This mode gives accurate results, but at the expense of time, it waits for numbers of the packets to calculate the connection record.

VFDT algorithm is used in both of the modes in order to perform the classification of connections, and then make the correct decision as to whether the connection is normal or attack. In this case, if there are attacks, it generates reports regarding the cause of the attacks, such as the IP address, MAC address, and time etc. The user can only continue to monitor the network and identify attacks at any time. The proposed system consists of four stages.

- Data Collection.
- Pre-Processing.
- Classification.
- Response.

3.3.1 Data Collection

This is the first stage in the proposed system. This part is concerned with capturing

the packets which passed through the whole network. Any packet that is targeted to any node in the network can be captured. In addition to a full capturing, the data and time fields are also displayed. These two values represent the data and the time that the packet was captured. All packets that have been captured will be monitored, and can be saved for analyzing.

Figure 3.4 shows the stages of packet decoder. The packet decoder takes packets from network interface via “Winpcap library”, and determines which protocol is in use for a given packet. Winpcap is a packet capture library under windows operating system and it is used to capture packets from the network. Sometimes, the packets which are taken by specific network interface using Winpcap library are also referred to as data acquisition or packet capture stage.

The packet decoder is actually a Series of decoders. Each sub-decoder decodes specific protocol elements starting from lower level data link protocols and moving up to transport layer. The packets move through the various protocol decoders to fill up a data structure with decoded packet data. After that, the packet stored in data structure is sent for pre-processing stage [23].

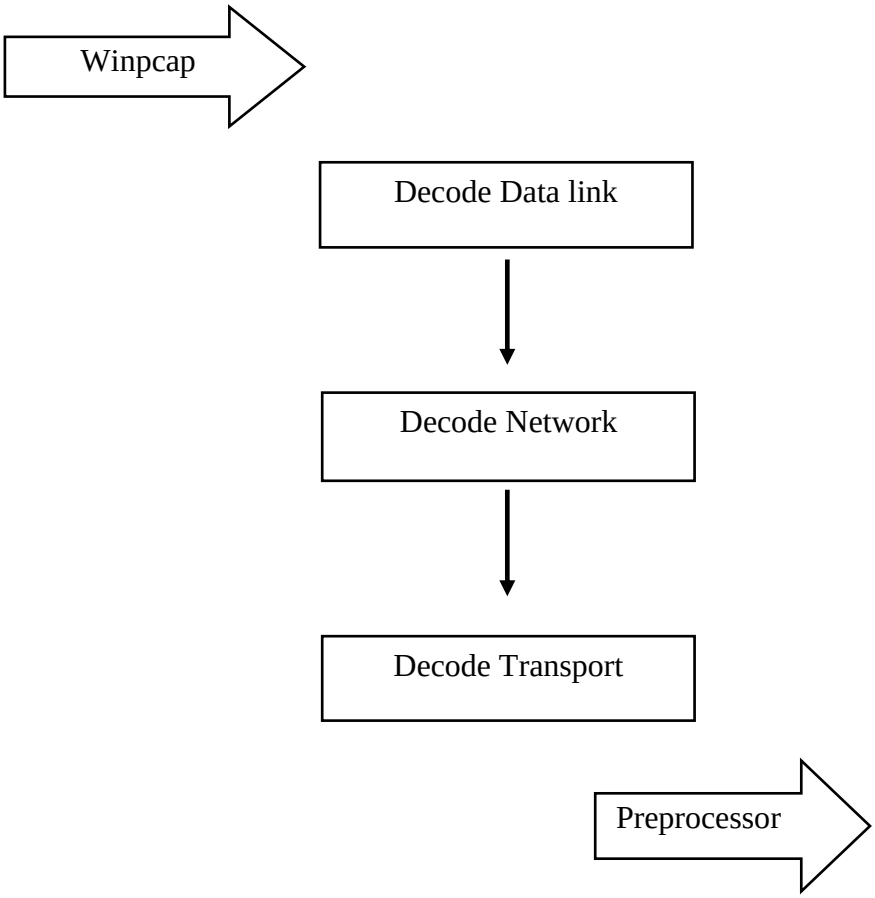

Figure 3.4: Stages of packet decoder [23].

3.3.2 Pre-processing

The Figure 3.5 shows the stages of pre-processing dataset before the training model. Data pre-processing is required in all knowledge discovery tasks, including network-based intrusion detection. Pre-processing refers to the process of extracting information about packet connection from header and data in addition to construction of new statistical features. Standard pre-processing steps include dataset creation, data cleaning, integration, and feature construction to derive new higher-level features, feature selection to choose the optimal subset of relevant features, reduction, and normalization. The most relevant steps for NIDS are now briefly described [24].

Data set Creation: It involves identifying representative network traffic for the training and the testing. These dataset “profiles” were created from several normal network sessions through weeks of normal work on the network. The profiles have been processed so as to get the values of the basic and statistical features, which are considered the normal and abnormal values for the network traffic.

Features Extraction: Detecting anomalous behavior depends on the values of features connection of network. This starts with features extraction, which takes the captured network packets as an input and then extracts features from these packets. It extracted basic feature from header of packets such as (protocol type, service, flag etc.) or extract content features from payload of packets such as (logged in, etc.) or compute the features manually “statistical” like (count, srv_count, etc).

Reduction: This is commonly used to decrease the dimensionality of the dataset by discarding any redundant or irrelevant features. This optimization is called feature selection.

Normalization: In this step, data samples containing both normal and numerical features were normalized and converted into linear discrete values “integers” in order to avoid the impact of overpowering the large scale features on the other features. As a result, the stages of pre-processing converts network traffic into a series of observations, where each observation is represented as a feature vector “connection”. Observations are optionally labeled with its class, such as “normal” or “anomalous”. These feature vectors are then suitable as input to data mining algorithms(i.e. VFDT).

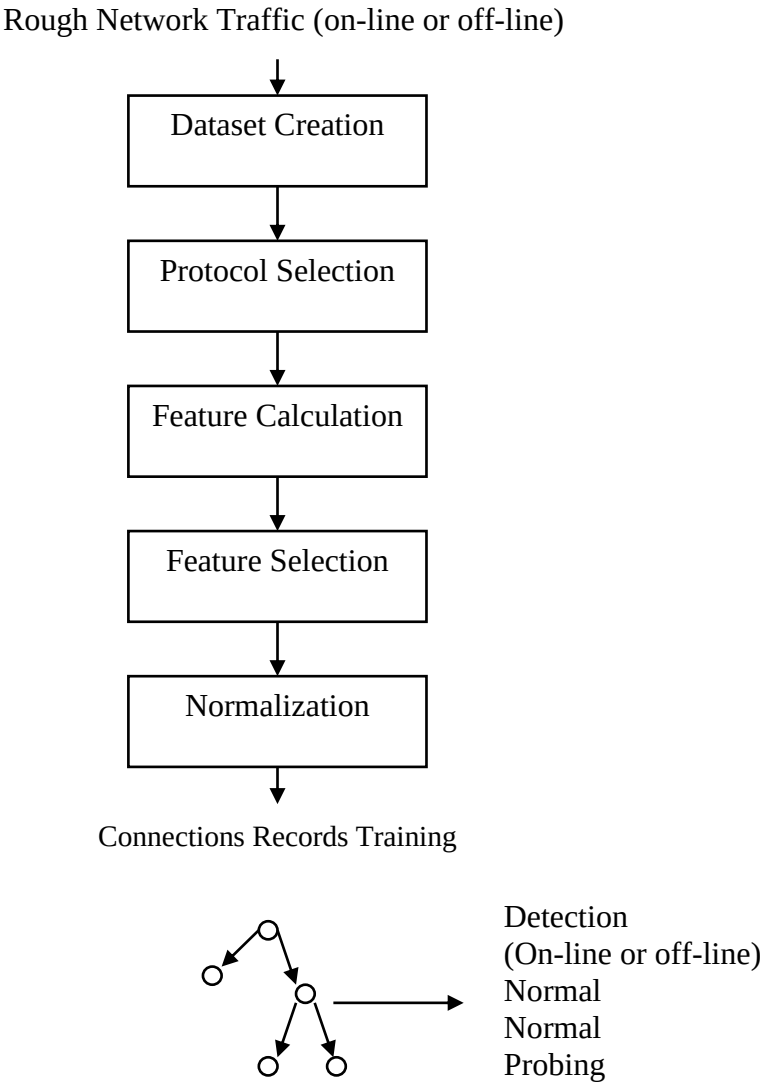

Figure 3.5: The stages of pre-processing data [24].

3.3.3 Classification

The purpose of this stage is to detect intrusions. It consists of abnormal detection mechanism also called abnormal detector. The structure of this abnormally detection is shown in Figure 3.6. We have two steps that are summarized as follows.

- The features extraction, which takes the captured network packets as an input and then extracts features from these packets.

- The main task of the detector is to identify intrusion patterns from through the extracted features by using VFDT algorithm.

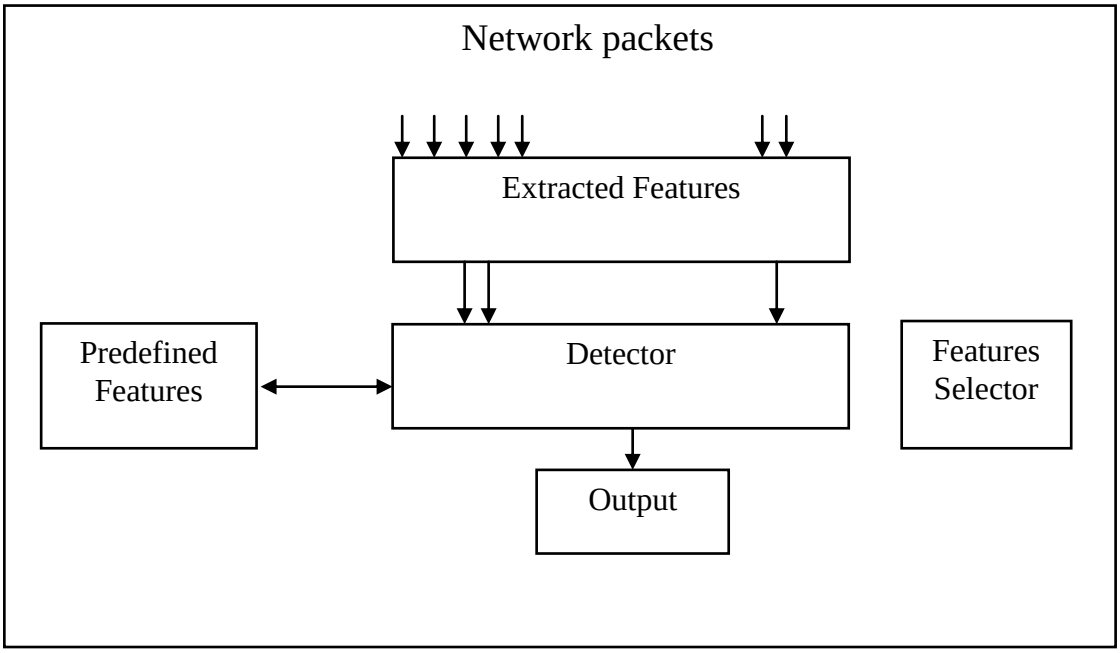

Figure 3.6: The structure of the anomaly detector [25].

3.3.4 Response

This is the last stage in the proposed system. Sometimes, this stage is called alerting model. This stage is concerned with deciding if an event or set of events is an intrusion or not.

It receives the output of anomaly detection and then gives the result in a report. This report shows if the event is an intrusion or it is normal. The report also specifies if it is the machine that causes the abnormal behavior, the data and its time. Then the IP address is updated and port number “attack signature” of the firewall or other nodes. When a node receives the attack signature, it checks if it exists in its hash table. If present, it means that the system is already alerted. If not, the attack signature is

added to the infected list. The updated attack signature is sent to all collaborating nodes, to prevent any damage that may be caused to the available services.

The report will show only the final results, which are the following fields.

Data and Time; Source IP address; Destination IP address; MAC Address

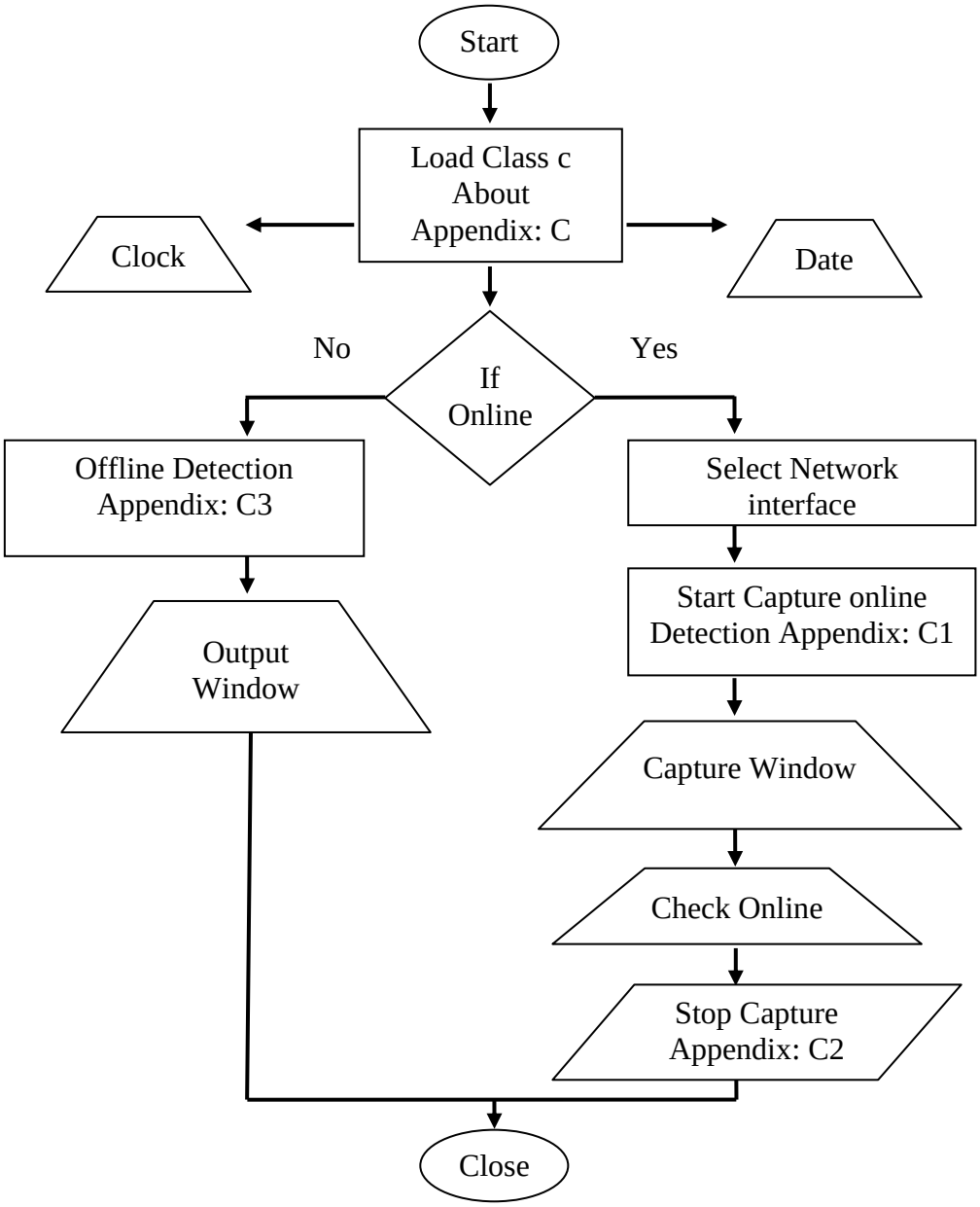

Figure 3.7: Block diagram of data of the proposed system.

Chapter 4

NADS SYSTEM IMPLEMENTATION AND RESULTS

4.1 Introduction

This chapter presents the implementation of the proposed system by using the algorithm that is described in chapters two and three, and it displays the main points that are used in the implementation process. The system is implemented on a computer connected to the internet and it can be applied on most networks. The visual C++ language version 6.0 was used to write the code of the system.

4.2 System Architecture

The proposed NADS system is shown in Figure 4.1. The system consists of the following parts.

1. Router, it is used for internet packets routing.
2. Switch-with-promiscuous mode which is used for packets switching, and traffic-sniffing for classification.
3. The NADS that are built to capture the packets, pre-processing, classify attacks and send alert reports.
4. Security analyst is to take a suitable action in regards to the alerts, and also to check if the alert is a true or false.
5. LAN. It contains three clients.

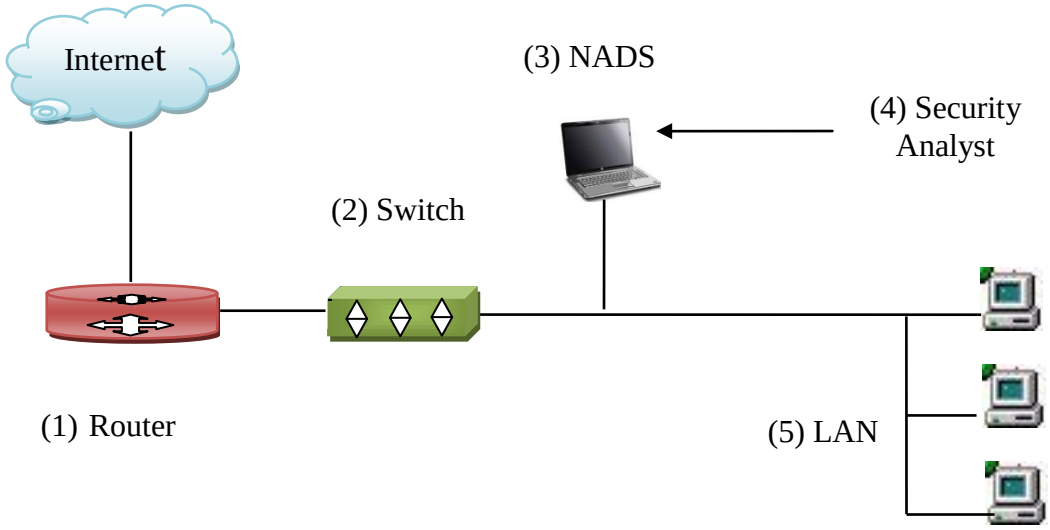

Figure 4.1: System architecture

The configuration of the system is done as follows:

1. Configure switch promiscuous mode.
2. Install the packet capture library on the NADS.
3. Install the classifier which belongs to the system.
4. Get internet connection by the router.

4.3 Performance Metrics

Confusion Matrix (CM) Table 4.1 is used in the measurement of performances of IDS and it has the following elements.

1. True Positive (TP): Number of connections that were correctly classified as attacks.
2. True Negative (TN): Number of connections that were correctly classified as normal.
3. False Positive (FP): Number of normal connections that were classified as attacks.
4. False Negative (FN): Number of attack connections that were classified as normal.

Table 4.1: Confusion Matrix (CM) [26].

	Normal	Intrusion (Attacks)
Normal	TN	FP
Intrusion (Attacks)	FN	TP

To rank the different results, there are standard metrics that have been developed for evaluating network intrusion detections. Detection Rate (DR) and False Alarm Rate (FAR) are the two most famous metrics that have already been used. DR or Percentage of Successful Predication (PSP) is computed as the ratio between the number of correctly detected attacks and the total number of attacks as in Equation (4.1).The FAR which is computed as the ratio between the number of normal connections, that are incorrectly misclassified as attacks and the total number of normal connections as in Equation (4.2) [26].

$$DR = \frac{TP}{TP + FN} \tag{4.1}$$

$$FAR = \frac{FP}{FP + TN} \tag{4.2}$$

Cost per Test (CPT) is computed by using the CM as in Table 4.2. The cost matrix calculates the cost of the confusion between the attacks and the normal connections. The visual C++ code is used to write our algorithm to calculate the cost of each test is called CPT as in Equation (4.3) [26].

$$CPT = \frac{1}{n} \sum_{i=1}^{5} \sum_{j=1}^{5} CMi, j \tag{4.3}$$

where *n*: is the number of instances in the test data set.

CM: denote the number of samples in class (*i*) misclassified as class (*j*).

Table 4.2: The Cost Matrix [26].

n		Normal	Probing	DoS	U2R	R2L
1	Normal	0	1	2	2	2
2	Probing	1	0	2	2	2
3	DoS	2	1	0	2	2
4	U2R	3	2	2	0	2
5	R2L	4	2	2	2	0

Table 4.2 shows the cost matrix which calculates the cost of the confusion between the attacks and the normal connections, and entry CM (*i*, *j*) represents the cost penalty for misclassifying an instance belonging to class (*i*) into class (*j)*.For the purpose of calculating the CPT.

Figure 4.2 shows the Receiver Operating Characteristic (ROC) which is a procedure, derived from statistical decision theory that was developed in the context of electronic signal detection. It is used to evaluate the predictive ability of classifiers. ROCs are plotted in coordinates which spanned by the rates of DR, and FAR classification. The ROC curve is a graph of sensitivity "y-axis" vs. specificity "x-axis". For example, maximizing sensitivity corresponds to some large (*y*) value on the ROC curve. Maximizing specificity corresponds to a small (*x*) value on the ROC curve. Thus a good first choice for a test cutoff value is that value which corresponds to a point on the ROC curve nearest to the upper left corner of the ROC graph [27].

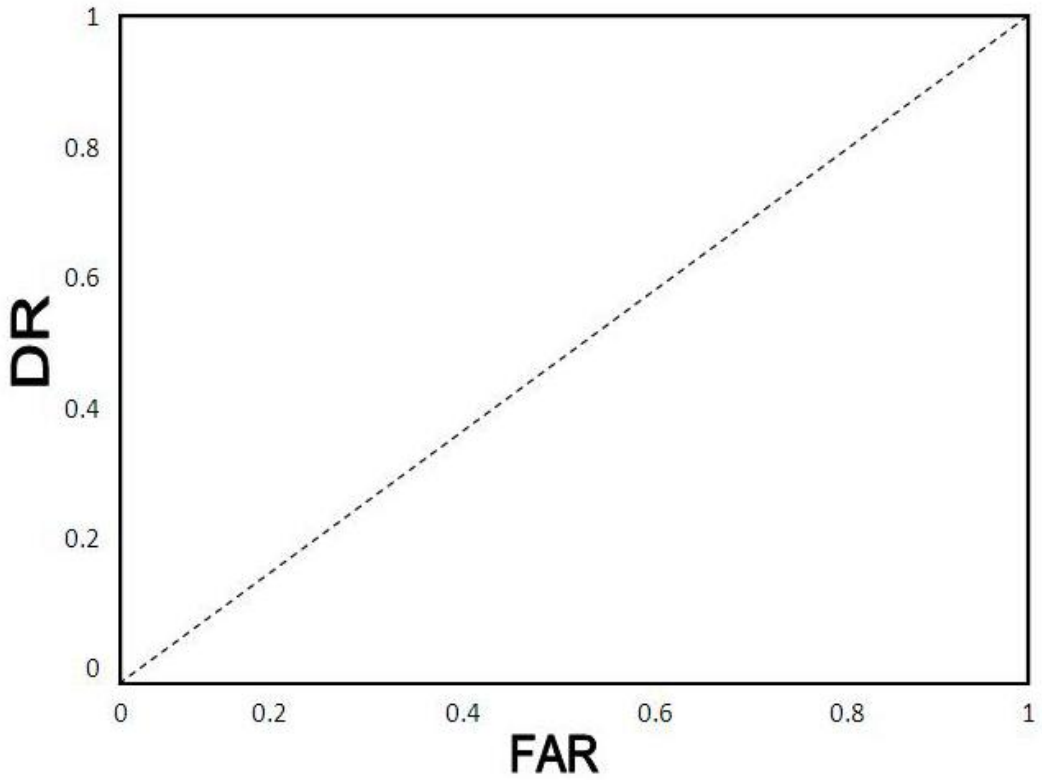

Figure 4.2: Receiver operating characteristic [27].

4.4 Real traffic Description

The packet capture is the first step of implementing the proposed system. It enables us to capture all packets of the real network traffic. This process runs in promiscuous mode, and captures all packets and stores them in data storage file. The data is stored as a set of traffic flows, with each instance being described by a set of features.

In Figure 4.3 the user can monitor the network traffic by pressing command box "start capture online detection". In this case, all packets that consist of the network traffic will be displayed.

The command box "stop capture" is used for stopping this process. The captured packets can be viewed on the screen or can be saved in a log file to be analyzed. The

offline detection button works to activate the system for detecting attacks in the offline mode.

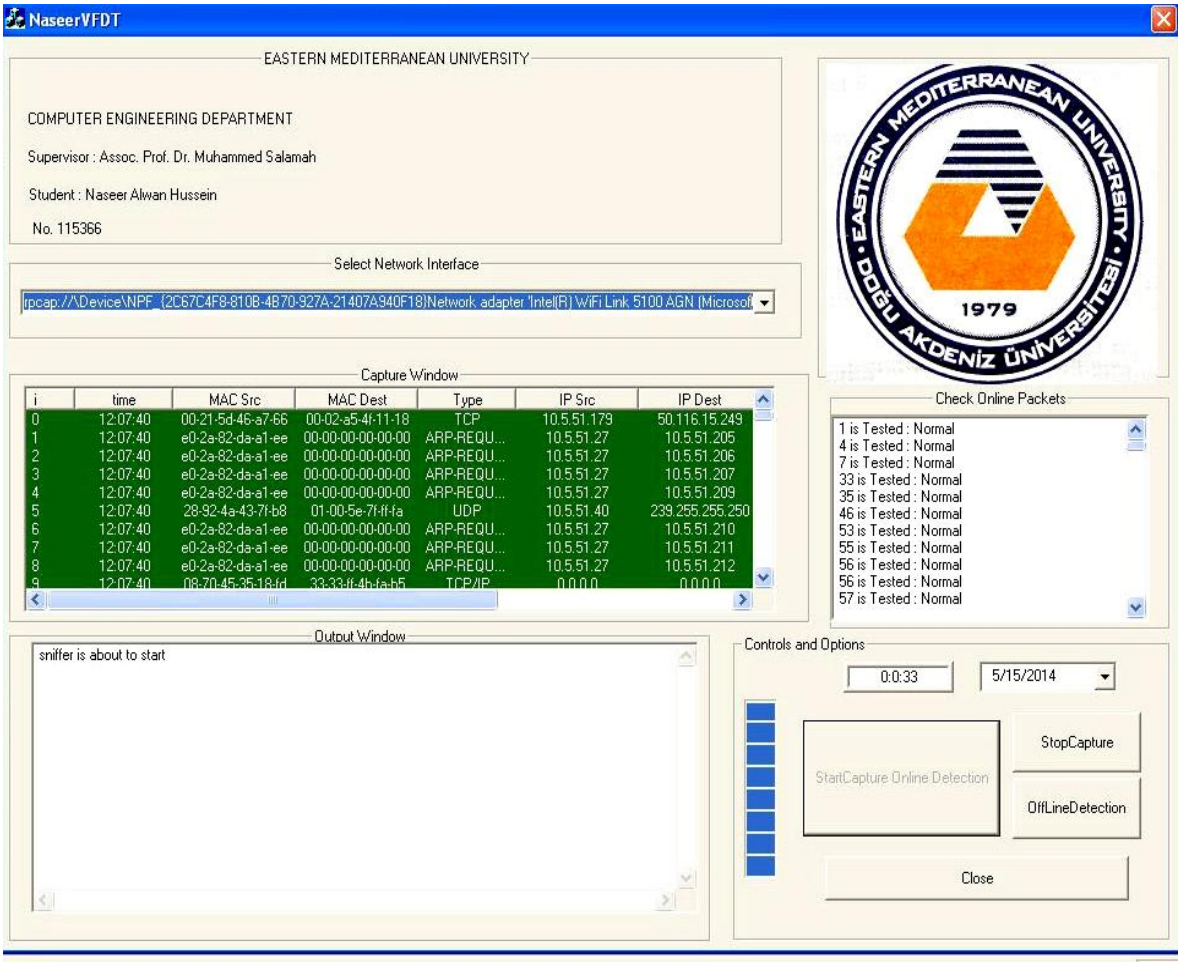

Figure 4.3: The windows that deal with real packet capturing

Figure 4.4 shows the real traffic of the network after saving in log file which contains the following fields: time, Type of service, Destination, src_byte, Dst_byte, Syn flag, Service, MAC addresses, IP source, and Flag.

i	Time	MAC Src	MAC Dest	Type	IP Src	IP Dest
1	20:19:05	f8-1a-67-dd-fc-50	01-00-5e-7f-ff-fa	UDP	192.168.0.1	239.255.255.250
2	20:19:05	f8-1a-67-dd-fc-50	01-00-5e-7f-ff-fa	UDP	192.168.0.1	239.255.255.250
3	20:19:05	00-21-5d-46-a7-66	01-00-5e-7f-ff-fa	ARP	192.168.0.2	239.255.255.250
4	20:19:05	00-21-5d-46-a7-66	01-00-5e-7f-ff-fa	ARP	192.168.0.8	239.255.255.250
5	20:19:05	00-21-5d-46-a7-66	01-00-5e-7f-ff-fa	ARP	192.168.0.3	239.255.255.250
6	20:19:05	00-21-5d-46-a7-66	01-00-5e-7f-ff-fa	ARP	192.168.0.0	239.255.255.250
7	21:34:22	00-21-5d-46-a7-30	f8-1a-67-dd-fc-5	TCP	192.168.0.8	239.255.255.253
8	21:34:22	00-21-5d-46-a7-30	f8-1a-67-dd-fc-5	TCP	192.168.0.0	239.255.255.253
9	21:34:23	00-21-5d-46-a7-30	f8-1a-67-dd-fc-5	TCP	192.168.0.7	239.255.255.250
10	21:24:01	00-21-5d-46-a7-30	f8-1a-67-dd-fc-5	TCP	192.168.0.3	239.255.255.252
11	21:24:01	00-21-5d-46-a7-30	f8-1a-67-dd-fc-b	UDP	192.168.0.0	239.255.255.257
12	21:24:02	00-21-5d-46-a7-30	f8-1a-67-dd-fc-b	UDP	192.168.0.0	239.255.255.257
13	21:24:02	00-21-5d-46-a7-30	f8-1a-67-dd-fc-5	TCP	192.168.0.0	239.255.255.257

Figure 4. 4: A sample of network traffic

A 1.3 MB data is collected within a week duration from a network that consists of 3 hosts. We used this data to extract the next connections and check.

4.4.1 Pre-processing Real Network Traffic

Real network traffic is captured and saved in the files. The system works to filter some packets and apply all steps of pre-processing which mentioned in the subsection 3.3.2. We wrote the different codes in visual C++ language which transforms the Tcpdump traffic into connection records with only 20 key features to detect attacks. Figure 4.5 shows the connection extraction of 20 features. These features are considered as an input vector to the VFDT algorithm for connection analysis, and then detect the attacks or anomalies.

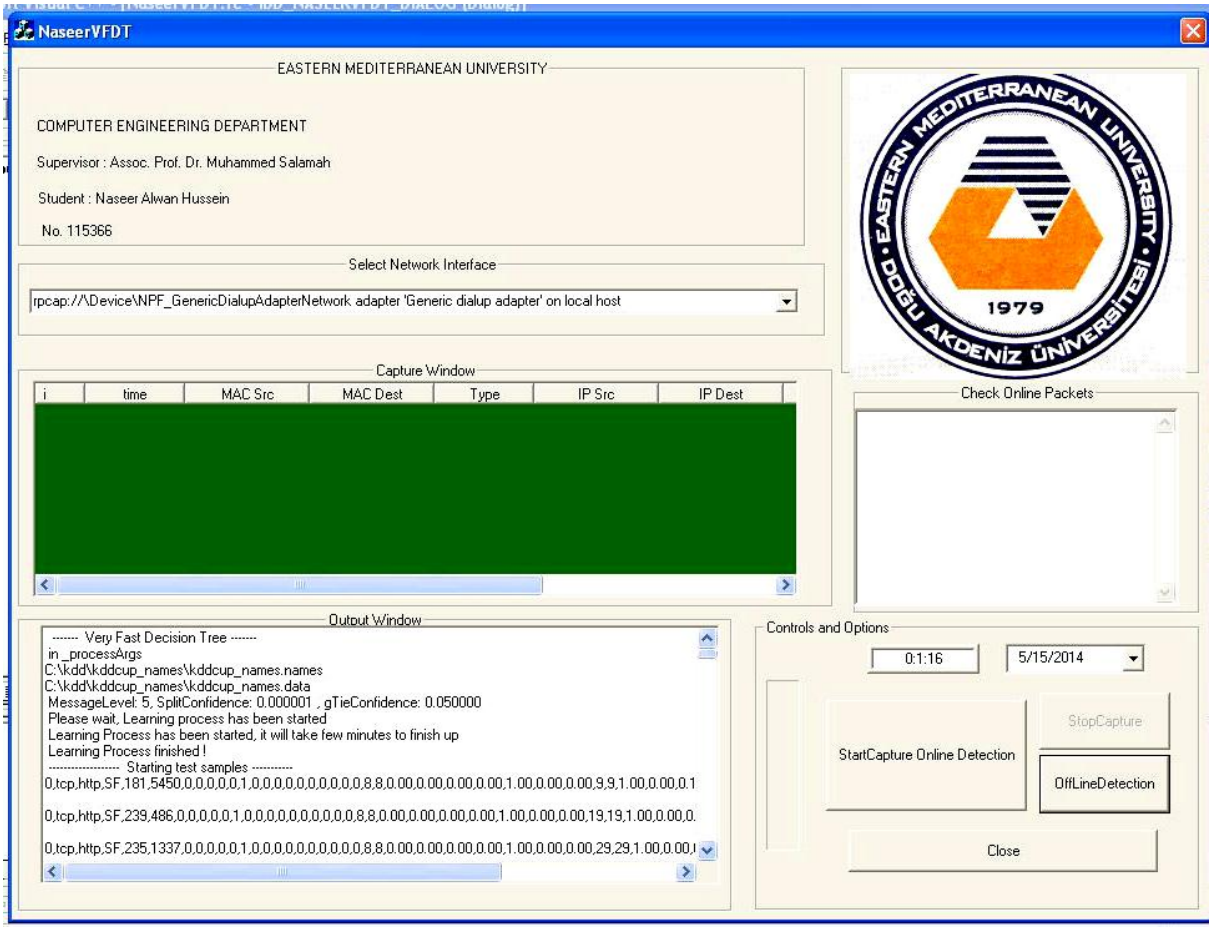

Figure 4.5: Extraction connection from real network traffic

4.5 Experimental Methodology and Results

In this section, we summarize our experimental results to detect the network intrusion by using the VFDT algorithm over KDDCUP99 data set. We first describe the data set used in the experiments and then discuss the results obtained. Finally, we evaluate our approach and compare the results with the results obtained by other researchers.

In the proposed NADS system, the first 20 features are used out of 41 features which are shown in Appendix B.

Figure 4.6 represents one sample connection of our model; these connections were

collected from the KDD CUP99 data set, which consists of “normal and abnormal behavior” for network traffic.

```
5,tcp,smtp,SF,959,337,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,1,0.00,00,0.00,0.00,1.00,0.
00.00,0.00,144,192,0.70,0.02,0.01,0.01,0.00,0.00,0.00,0.00,normal.
```

Figure 4.6: A sample connection in the network

VFDT algorithm is used to distinguish attacks from normal behaviors and identifies different types of intrusions as described in section 2.5. The VFDT is run in the offline mode. KDD CUP 99 dataset is used in the training phase. The VFDT algorithm must undergo the training and testing phases before it is used as a detector.

We carried out four experiments on KDD CUP99 dataset in order to get the best results. The conditions of these experiments are as follows.

Experiment 1: We used the whole of KDD CUP99 dataset for training, but we selected 20 features only.

Experiment 2: We used the whole KDD CUP99 dataset for training, but we choose 15 features.

Experiment 3: We used 10% of the KDD CUP99 dataset for training, and we selected 41 features.

Experiment 4: We used the whole KDD CUP99 dataset for training, but we choose only 5 the basic features.

For the attacks DoS, probe, U2R, and R2L, the True Positive percentage (TP %) is defined as the number of the related attack TP's over the total number of TP's.

The results of these experiments can be seen in Table 4.3.

The first experimental result indicated that the proposed NADS system achieved a DR percent of 93.825 % for attacks by using the VFDT algorithm, whereas the achieved FAR rate is 0.608%. In addition, the TP% for DoS, Probe, U2R, and R2L is 93.106%, 79.04%, 22.84%, and 36.64% respectively.

The result of the second experiment is close to the first experiment, although the detection rate for U2R becomes zero.

The third experimental result indicated that the proposed NADS system achieved a DR percent of 95.532 % and a FAR percent of 0.993%. In addition, the TP% for DoS, Probe, U2R, and R2L is 96.4%, 24.2%, 75.6%, and 81.6% respectively.

The fourth experimental result indicated that the proposed NADS system achieved a DR percent of 90.34 % and a FAR rate of 0.78%. In addition, the TP% for DoS, Probe, U2R, and R2L is 92.67%, 75.4%, 0%, and31.32% respectively.

Table 4.3 Performance metric between different experiments

Experiments	DR%	FAR%	DoS TP %	Probe TP %	U2R TP %	R2L TP %
Experiment1	93.825	0.608	93.106	79.04	22.84	36.64
Experiment2	92.56	0.720	88.1	80	0	33.6
Experiment3	95.532	0.993	96.4	24.2	75.6	81.6
Experiment4	90.34	0.78	92.67	75.4	0	31.32

The following parameters are discussed to clarify our results precisely in terms of system accuracy, classification speed, and memory allocation.

4.5.1 System Accuracy Results

The VFDT algorithm has better accuracy performance when compared with other classification algorithms, where its classification accuracy is 93.825%. The ROC curve of the system is shown in Figure 4.7. The goal from our research is to detect many attacks while minimizing the generation of FAR. In addition Figure 4.7 shows that our system is able to detect most of the attacks for the KDD CUP99 data set at a low FAR rate of 0.608%.

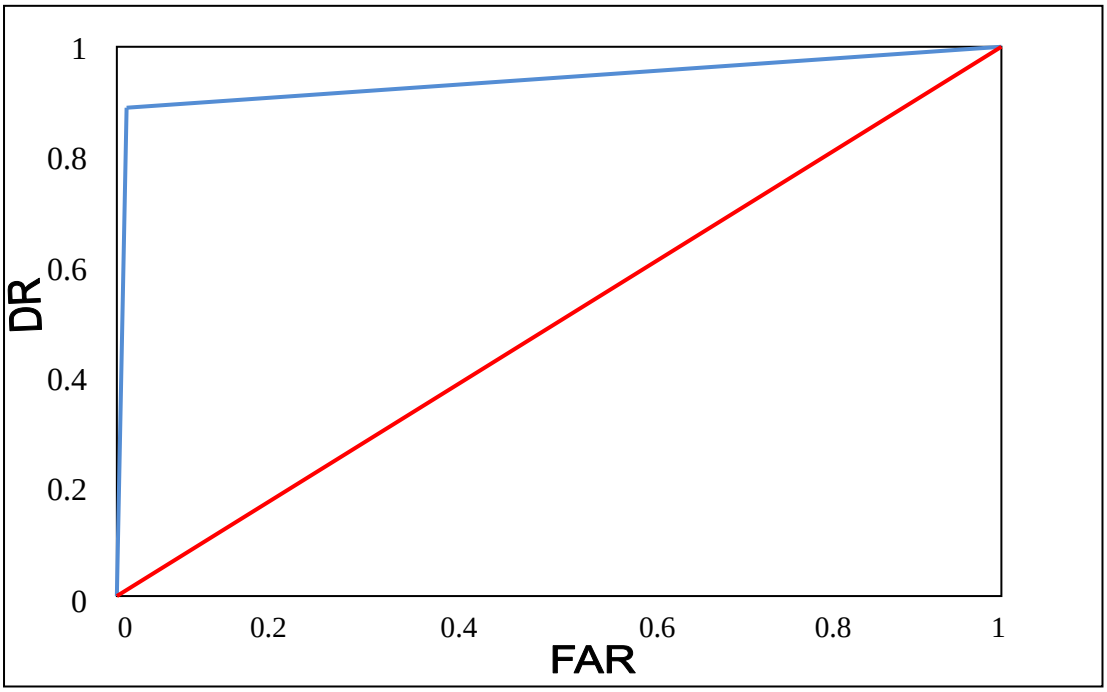

Figure 4.7: ROC curve for VFDT algorithm.

4.5.2 Classification Speed Results

Figure 4.8 shows the training time versus number of connections. The VFDT algorithm uses 100,000 connections at time 4.09s, whereas it uses1000, 000 connections at time 38.97s.

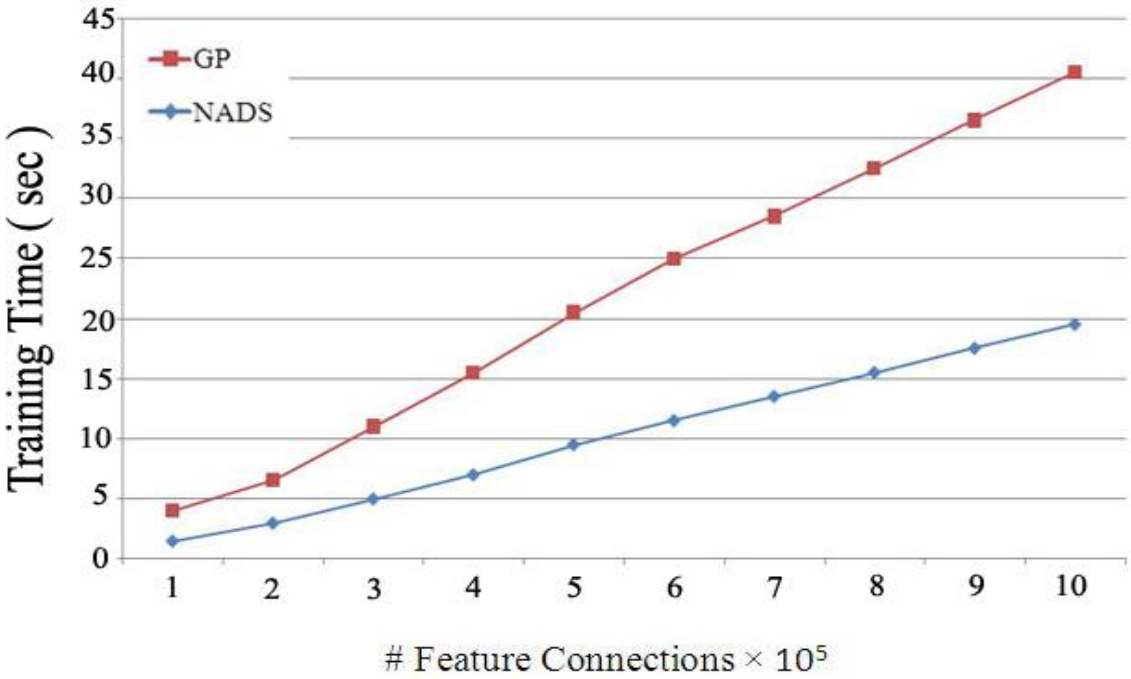

Figure 4.8: Training time versus number of connections.

4.5.3 Memory Allocation Results

Figure 4.9 shows the number of nodes which are created when we use the VFDT algorithm. The unexpected increasing at the beginning of the chart represents the beginning of the construction of a tree. The sufficient number of connections is collected to be able to classify all types of attacks in the same tree. It continues to increase gradually; while the number of nodes is almost stable.

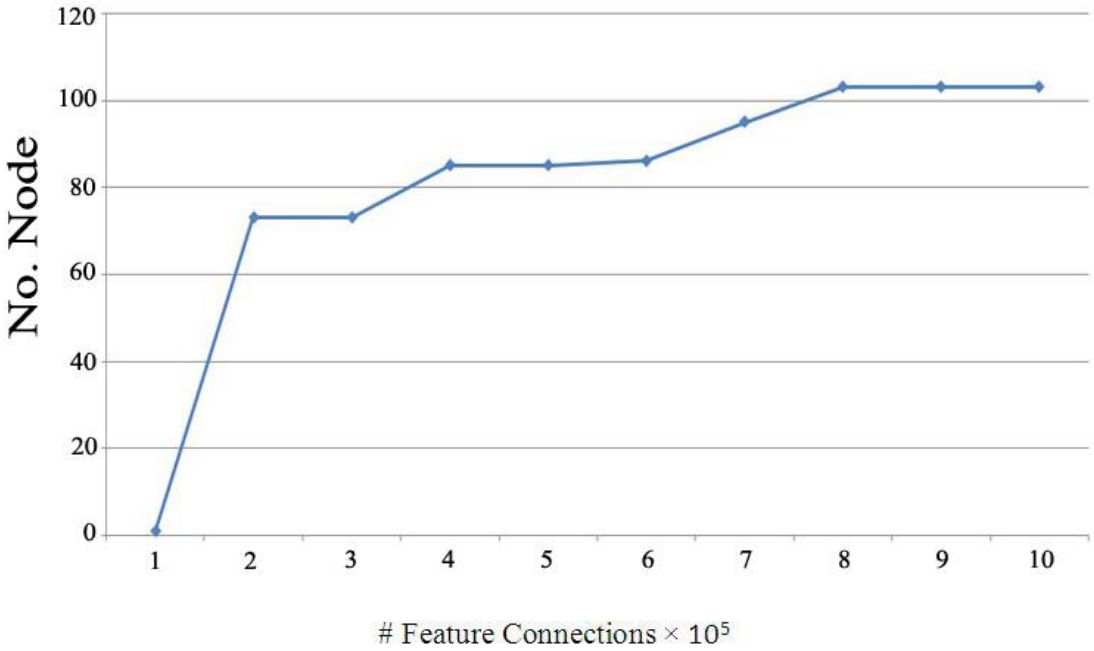

Figure 4.9: No of nodes versus number of connections.

4.6 Comparison of the proposed NADS system using the VFDT algorithm with Other Algorithms.

The results are discussed for each attack in the previous sections. In this section, the detection rate and speed of classification for the proposed NADS system using the VFDT algorithm are compared with several classification algorithms.

Table 4.4 shows the Detection Rate (DR), Training Time (TT) per each example for the proposed NADS system using the VFDT algorithm and other classification algorithms. The table also shows the different numbers of examples used in both the training phase and the testing phase of the classification algorithms.

The number of examples is an influential factor on the percentage of the classification accuracy and the training time. The proposed NADS system using the VFDT algorithm outperforms all algorithms in terms of training time per example since it has the lowest time. With respect to DR% performance, the proposed NADS

system using the VFDT algorithm outperforms all other algorithms except the Genetic Programming (GP) algorithm where it has a higher DR rate of 98%. This is because the GP algorithm uses 41 features in the training phase. The proposed NADS system achieves better performance for the anomaly detection at high speed.

Table 4.4: A comparison of proposed system with other classification algorithms that used KDD CUP 99 data set

No.	Algorithm	Training Datasets Size	Testing Datasets Size	DR%	FAR%	Training Time (TT)in sec.	Training Time Per Example
1	[NADS]	1,074,985	67,688	93.825	0.608	39.88	0.000003
2	[GP]	24,780	311,028	98	0.739	6480	0.2615
3	[SVM]	1,132,365	73,247	57.6	0.89	1040.4	0.0009
4	[SOM]	49,596	15,437	91.65	0.933	192.16	0.0038
5	[MLP]	49,596	15,437	92.03	0.91	350.15	0.0070
6	[BN]	49,596	15,437	90.62	0.897	6.28	0.0001
7	[J48]	49,596	15,437	92.06	0.62	15.85	0.0003
8	[LBK]	49,596	15,437	92.22	0.635	10.63	0.0002
9	[Apriori]	444,458	49,384	87.5	0.82	18.94	0.00004
10	[ITI]	169,000	311,029	92.38	0.598	17	0.0001
11	[PART]	444,458	49,384	46.67	0.935	48.8	0.0001
12	[RF]	65,525	65,525	90.08	0.781	129	0.0019
13	[K-Means]	55,000	25,000	86	0.653	13	0.0002
14	[FL]	54,226	56,226	91.25	0.721	87.9	0.0016
15	[ANN]	4,947	3,117	92.268	0.561	780	0.1576
16	[C4.5]	49,596	15,437	92.06	0.582	15.85	0.0003

Chapter 5

CONCLUSION

The primary aim of this thesis is to propose a Network-based Anomaly Detection System (NADS) which helps to take insidious attacks under control. This technique depends on the quality or fastness of network, that is to say, it is expected that it will distinguish every abnormal behavior from normal. The major goal of anomaly observation system is to expose the detectable and undetectable intrusions. This technique works on two modes, on-line and offline modes. The proposed NADS system uses the Very Fast Decision Tree algorithm (VFDT), and it classifies the connections as normal or abnormal.

The experimental results indicated that the proposed NADS system achieved a high classification accuracy rate of 93% by using the VFDT algorithm. It is the highest performance compared with all other algorithm except the Genetic Programming (GP) algorithm where it has a higher DR rate of 98%. The speed of training "building and testing" didn't exceed 39.88 seconds by using the VFDT algorithm, whereas it is in terms of hours for others systems.

The VFDT outperforms all other algorithms in terms of training time per example since it has the lowest time. Many researches related to Intrusion Detection System (IDS) are being introduced recently. The research that can achieve a good score and

add an important contribution is the research that solves the important problematic issues. Those include the detection optimization, and data security.

As a future work, this study can be extended in different aspects such as.

- The proposed NADS system used the VFDT algorithm for detection. Hence, it can be analyzed using other detection algorithms as well.

- Also, one can try to solve the overlaps problem between normal and U2R classes, and determine the class of each one.

- As the proposed NADS system uses “anomaly detection technique”, it can be modified to use the misuse technique as well.

- Also, the proposed NADS system can be experimented on other data set like SSENet-2011, and UNB ISCX 2012.

REFERENCES

[1] K. Sandeep, "Classification and Detection of Computer Intrusions", PhD Thesis, Purdue University, August 1995.

[2] A. Ali, W. L. Mahbod, "Network Intrusion Detection and Prevention, Concepts and Techniques", Springer Media, New York LLC, 2010.

[3] P. Animesh, P. Jung-Min, " An Overview of Anomaly Detection Techniques: Exiting Solutions and Latest Technological Trend", The international Journal of Computer and Telecommunications Networking Vol.51, issue 12, pp. 3448-3470,2007.

[4] G. A. Blank, "TCP/IP Jumpstart: Internet Protocol Basics", 2nd Edition, John Wiley and Sons, 2002.

[5] H. O. Alanazi, R. Noor, B. Zaidan, A. Zaidan," Intrusion Detection System: Overview", Journal of Computing, Vol. 2, Issue 2, pp.32-48, February 2010.

[6] L. Theodor's, P. Konstantinos, "Data Mining Techniques for Network Intrusion Detection Systems," Department of Computer Science and Engineering UC Riverside, Riverside CA, 2004.

[7] Z. Zhi-Hua, L. Hang, Y. Qiang, "Advances is Knowledge Discovery and Data Mining," 11th Pacific-Asia Conference, PAKDD, Springer, China, Vol.4426, 2007.

[8] L. Wenke, J. Salvatore, W. Mok, “A Data Mining Framework for Building Intrusion Detection Models, “IEEE Symposium on Security and Privacy, pp. 120-132, 1999.

[9] T. Pang-Ning, S. Michael, K. Vipin, “Introduction to Data Mining”, 2nd Edition, Addison Wesley, March 2006.

[10] M. M. Gaber,” Advances in Data Stream Mining”, John Wiley and Sons, Vol. 2, No. 1, pp. 79-85, 2012.

[11] T. Mahbod, E. Bagheri, A. Ghorbani, “A Detailed Analysis of the KDD CUP99 Data set”, 2nd IEEE Symposium on Computational Intelligence for Security and Defense Application (CISDA), Vol. 3, No. 5,pp. 322-332,2009.

[12] M. Tasuya, N. Ayahiko, “ Detection of Fraud use of Credit Card by Extended VFDT ,” IEEE Transactions on Knowledge and Data Engineering(TKDE), Vol. 21, No. 11, pp. 1505-1514,2011.

[13] M. Tatsuya, I. Masayuki, N. Ayahiko, K. Osmu, “Extension of Decision Tree Algorithm for Stream Data Mining Using Real Data,” 5th International Workshop on Computational Intelligence and Applications (IWCIA). IEEE Systems, pp. 208-212, 2009.

[14] B. Daniel, J. Couto, S. Jajodia, N. Wu, “ADAM: A Test bed for Exploring the use of Data Mining in Intrusion Detection”, SIGMOD, Vol. 30, No. 4, 2001.

[15] H. Sang-Jun, C. Sung-Bae, "Combining Multiple Host-Based Detectors using Decision Tree," 16th Australian Conference on Artificial Intelligence, Springer, Vol. 2903, pp.208-220, 2003.

[16] S. Ahmed, "Intrusion Detection System using Data Mining", Applied Science Department, Master Thesis, University of Technology, 2006.

[17] B. Yacine, C. Frederic, "Neural Networks vs. Decision Tree for Intrusion Detection," IEEE/IST Workshop on Monitoring Attack Detection and Mitigation (MonAM), 2006.

[18] P. Mrutyunjaya, R. P. Manas," Network Intrusion Detection using Naïve Bayes", International Journal of Wei-computer Science and Network Security (IJCSNS), Vol. 7, No.12, pp. 455-464, December 2007.

[19] P. Mrutyunjaya, R. P. Manas," Evaluating Machine Learning Algorithms for Detecting Network Intrusion," International Journal of Recent Trends in Engineering Vol. 1, No. 1, May 2009.

[20] A. N. Huy, D. Choi, " Application of Data Mining to Network Intrusion Detection: Classifier Selection Model", Asia-pacific Network Operation and Management Symposium (APNOMS), Springer-Verlag Berlin, Heidelberg, pp. 399-406, 2008.

[21] M. Dewan, H. Nouria, B. Emna, Z. Mohammed, M. Chowdhury, " Attacks Classification in Adaptive Intrusion Detection using Decision Tree", World Academy of Since, Engineering and Technology, No.63, pp.27-44, 2010a.

[22] M. Dewan, H. Nouria, Z. Mohammad, " Combining Naïve Bayes and Decision Tree for Adaptive Intrusion Detection", International Journal of Network Security and its Applications(IJNSA), Vol. 2, No. 2,pp. 233-247, 2010b.

[23] R. Hanumantha, G. Srinivas, D. Ankam, K. Vikas, " Implementation of Anomaly Detection Technique using Machine Learning Algorithms", International Journal of Computer Science and Telecommunications (IJCST), Vol. 2, Issue 3, pp. 25-31,2011.

[24] R. Shanmugavadiru, N. Nagarajan, " Network Intrusion Detection System using Fuzzy Logic," Indian Journal of Computer Science and Engineering (IJCSE), Vol. 2, No. 1, pp. 101-111,2011.

[25] G. Radhika, S. Anjali, C. J. Ramesh, "Parallel Misuse and Anomaly Detection Model," International Journal of Network Security, Vol.14, No.4, pp. 211-225, July 2012.

[26] S. William, "Network Security Essentials: Applications and Standards," 1st Edition, Prentice-Hall PTR Upper Saddle River, USA, 1999.

[27] E. Terry, “Intrusion Detection: Network Security Beyond the Firewall,” John Wiley and Sons, New York, USA, 1998.

[28] F. Behrouz, “TCP/IP Protocol Suite”, 3rd Edition, Mc Grawa Hill Education, 2005.

[29] K. Urupoj, S. Surasak, “Network-based Intrusion Detection Model for Detecting TCP,” Department of Computer Engineering, Kasetsart University, Bangkok, Thailand, 2000.

[30] T. Jawin, “Network Protocols Handbook”, 2nd Edition, Jawin Technologies, 2005.

[31] T. Lammle, “CCNA Cisco Certified Network Associate Study Guide”, 6th Edition, Wiley Publishing, 2007.

[32] S. Panear, S. Mao, J. Ryoo, Y. Li, “Essentials A Lab-based Approach,” Cambridge University Press, 2004.

[33] C. Hunt, “TCP/IP Network Administration”, 3rd Edition, Networking OReilly Media, 2002.

[34] P. Sandhye, A. Ajith, T. Johnson, “ Intrusion Detection Systems using Decision Trees and Support Vector Machines”, International Journal of Applied Science and Computations, Vol. 11,No.3, pp.118-134, 2002.

[35] G. Meera, S. Srivatsa, “Detecting and Preventing Attacks using Network Intrusion Detection Systems”, International Journal of Computer Science and

Security, Vol. 2, issue 1, pp. 295-302, 2005.

[36] C. Endrof, E. Schultz, J. Mellaner, "Intrusion Detection and Prevention", California, McGeaw-Hill.2004.

[37] D. Dorothy, " An Intrusion-detection Model", IEEE Transactions on Software Engineering Vol. 13, No. 2, pp. 222-232, 1978.

[38] D. Herve, D. Mare, W. Andreas, "Towards A taxonomy of Intrusion Detection Systems", Computer Networks, No. 8, pp. 805-822, 1999.

[39] M. A. Rassam, " Anomaly Intrusion Detection System using Immune Network with Reduced Network Traffic Features," Master Thesis, Faculty of Computer Science and Information Systems, University Technology, Malaysia, 2010.

[40] H. Marko, "Traffic Analysis for Intrusion Detection in Telecommunications Networks", Master Thesis, Computing and Electrical Engineering Faculty Council, 2011.

[41] C. Brenton, C. Hunt, "Mastering Network Security", 2nd Edition, SYBEX in Alameda CA, USA, 2003.

[42] J. Menga, C. Timm, "CCSP Secure Intrusion Detection and SAFE Implementation Study Guide", John Wiley and Sons, SYBEX, 2004.

[43] A. Lazarevic, "Data Mining for Intrusion Detection Encyclopedia of Data Warehousing and Mining", Idea Group, 2005.

[44] E. Levant, L. Aleksandra, "The MINDS-Minnesota Intrusion Detection System

“, IDEAS, University of Minnesota, ACM New York, pp.255-261, 2003.

[45] N. A. Matthew, S. G. Shiva, “Comparative Analysis of Serial Decision Tree Classification Algorithms”, International Journal of Computer Science and Security (IJCSS), Vol. 3, issue 3, pp. 230-240, 2009.

[46] D. Pedro, H. Geoff, ”Mining High Speed Data Streams”, ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 71-80, 2000.

[47] W. XU, Z. QIN, “ Constructing Decision Tree for Mining High-Speed Data Streams”, Chinese Journal of Electronics, Vol. 21, No. 2, pp. 215-220, April 2012.

[48] Y. Hang, F. Simon, “ Moderated VFDT in Stream Mining using Adaptive Tie Threshold and Incremental Pruning”, Springer-Verlag Berlin, Heidelberg, pp. 471-483, 2011.

[49] B. Salem, S. Karima, T. Karim, “Preprocessing Rough Network Traffic for Intrusion Detection Purposes,” International Telecommunications IADIS, Networks and Systems, pp. 105-109, 2007.

[50] J. D. Jonathan, J. C. Andrew, “Data Preprocessing for Anomaly based Network Intrusion Detection: Review”, Journal Computer and Security, Vol. 30, No. 67, 353-375, 2011.

[51] N. M. Fatin, S. K. Aman, “ Identifying False Alarm Rates for Intrusion Detection System with Data Mining”, International Journal of Computer Science and Network Security(IJCSNS), Vol. 11, No. 4, pp. 22-28, 2011.

[52] K. Bhati, S. Shukla, S. Jain, "Intrusion Detection using Clustering", International Journal of Computer Applications, Vol. 1, Issue 2,pp. 125-138, 2010.

[53] S. J. Stolfo, W. Fan, W. Lee, A. Prodromidis, P. K. chan, "Costbased Modeling for Fraud and Intrusion Detection: Results from the Jam Project," Discex, Vol. 02, pp.1130, 2000.

[54] B. Marjan, E. Salahi, M. Khaleghi, " An Improved Intrusion Detection Technique Based on Two Strategies using Decision Tree and Neural Network", JCIT Journal, Vol. 4, No. 4, pp. 96-101,2009.

[55] V. Podgorelec, P. Kokol, B. Stiglic, I. Rozman, " Decision Trees: An Overview and Their use in Medicine," Journal of Medical Systems Kluwer Academic/ Plenum Presss, Vol. 26, No. 5, pp. 445-463, 2002.

[56] M. F. Kamel, B. Aoued, "Securing Network Traffic using Genetically Evolved Transformations", Malaysian Journal of Computer Science, Vol. 19, pp. 3-23, 2006.

[57] KDD CUP 99 Intrusion Detection Data

http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html., 1999.

[58] J. Koziol, "Intrusion Detection with Snort", 2nd Edition, Sams Indianapolis, USA, 2003.

APPENDICES

Appendix: A

List features KDD CUP 99 dataset for which the class is selected most relevant features [39].

No.	Class Label	Relevant Features
1	Back	5,6
2	Land	7
3	Neptune	3,4,5,23,26,29,30,31,32,34,36,37,38,39
4	Pod	8,24
5	Smurt	2,3,5,6,12,25,29,30,32,36,37,39
6	Teardrop	8
7	Satan	27
8	Ipsweep	36,24
9	Nmap	5
10	Portsweep	28
11	Normal	3,6,12,23,25,26,29,30,33,34,35,36,37,38,39
12	Guess_passwd	11,6,3,4
13	Ftp_writ	9,23
14	Imap	3,39
15	Phf	6,10,14,5
16	Multihop	23
17	Warezmaster	24,6,1
18	Warezclient	3,24,26
19	Spy	39,1
20	Buffer_overflow	3,24,14,6
21	Loadmodule	36,24,3
22	Perl	14,16,18,5
23	Rootkit	24,23,3

Appendix: B

Attributes description of KDD CUP 99 dataset [54].

No.	Network attributes	Description	Type
1	Duration	Duration of the connection in second.	Continuous
2	Protocol_type	Connection protocol (e.g. TCP, UDP, ICMP).	Discrete
3	Service	Destination service (e.g. telnet, ftp, http, pop3…)	Discrete
4	Flag	Status flag of the connection (e.g. REJ, SF, S0…)	Discrete
5	Src_bytes	Bytes sent from source to destination	Continuous
6	dst_bytes	Bytes sent from destination to source	Continuous
7	Land	1 if connection is from /to the same host/port; 0 otherwise	Discrete
8	Wrong_fragment	Number of wrong fragments	Continuous
9	Urgent	Number of urgent packets	Continuous
10	Hot	Number of”hot” indicators	Continuous
11	Num_failed_logins	Number of failed logins	Continuous
12	Logged_in	1 if successfully logged in; 0 otherwise	Discrete
13	Num_compromised	Number of “compromised” conditions	Continuous
14	Root_shell	1 if root shell is obtained; 0 otherwise	Continuous
15	Su_attempted	1 if ”as root” command attempted; 0 otherwise	Continuous

No.	Network attributes	Description	Type
16	Num_root	Number of root accesses	Continuous
17	Num_file_creations	Number of file creation operations	Continuous
18	Num_shells	Number of shell prompts	Continuous
19	Num_access_files	Number of operations on access control files	Continuous
20	Num_outbound_cmds	Number of outbound commands in an ftp session	Continuous
21	Is_host_login	1 if the login belongs to the “hot” list; 0 otherwise	Discrete
22	Is_guest_login	1 if the login is a “ guest” login; 0 otherwise	Discrete
23	count	Number of connections to the same host as the current connection in the past two seconds	Continuous
24	Srv_ count	Number of connections to the same service as current connection in the past two seconds	Continuous
25	Serror_rate	No. of connections that have “ SYN” errors	Continuous
26	Srv_ Serror_rate	No. of connections that have “ SYN” errors	Continuous
27	Rerror_rate	No. of connections that have “ REJ” errors	Continuous
28	Srv_ rerror_rate	No. of connections that have “ REJ” errors	Continuous
29	Same_ rerror_rate	No. of connections to the same service	Continuous
30	Diff_ Srv_ rate	No. of connections to different services	Continuous
31	Srv_ Diff_host_ rate	No. of connections to different hosts	Continuous
32	Dst_ host_count	Count of connections having the same destination host	Continuous
33	Dst_ host_ Srv_count	Count of connections having the same destination host and using the same service	Continuous

No.	Network attributes	Description	Type
34	Dst_ host_ Same_ rate	Count of connections having the same destination host and using the same service	Continuous
35	Dst_ host_Diff_ Srv _ rate	No. of different services on the current host	Continuous
36	Dst_ host_ Same_src_port_ rate	No. of connections to current host having the same src port	Continuous
37	Dst_ host_ Srv _ Diff_ host_ rate	No. of connections to same service coming from different hosts	Continuous
38	Dst_ host_ serror_ rate	No. of connections to the current host that have an S0 error	Continuous
39	Dst_ host_ Srv _serror_ rate	No. of connections to the current host and specified service that have an S0 error	Continuous
40	Dst_ host_rerror_ rate	No. of connections to the current host that have an RST error	Continuous
41	Dst_ host_ rerror_ rate	No. of connections to the current host and specified service that have an RST error	Continuous

Appendix: C

Figure C1: Start Capture (Online Detection).

```
void C Naseer VFDT Dlg::On Start()

{

// TODO: Add your control notification handler code here

// TODO: Start capturing thread

if (is Nic O penned){

 Print Out(" sniffer is about to start ");

HANDLE thd1 = Create Thread
(NULL,0,(LPTHREAD_START_ROUTINE)sniff Thread,(LPVOID) this,0,&tid);

C W nd * ctrl = this->Get Dlg Item (IDC_START);

ctrl->Enable Window(false);

ctrl = this->Get Dlg Item(IDC_STOP);

ctrl->Enable Window(true);

       }

Else {

 Message Box (" NIC has not been o penned or Selected ");

           }

}
```

```
void C Naseer VFDT Dlg::On Sel change Nic Combo()

{

// TODO: Add your control notification handler code here

int selection  =  m Nic Combo. Get Cur Sel ();

char err buf [PCAP_ERRBUF_SIZE];

p cap _ if _ t *d;

d=all devs;

for (int i=0;i<selection; i++) d=d->next;

if (d->addresses->add r->sa _family==AF_INET)

{

If Info. Device Name =d->name;

If Info. Description=d->description;

If Info. IP Add r=(((struct  sock add r_ in *)d->addresses->add r)->sin _add r. s _add r);

If Info. IP Mask=(((struct sock add r_ in *)d->addresses->net mask)->sin _add r. s_ add r);

//Flag=true;     //

}
```

```
if ((If Info. Ad handle = pcap_open (If Info. Device Name, // name of the device

65536, // portion of the packet to capture.

// 65536 grants that the whole packet will be captured on all the MACs.

PCAP_OPENFLAG_PROMISCUOUS,// promiscuous mode (nonzero means promiscuous)

1000, / read timeout

NULL,

Err buf // error buffer)) == NULL)

{

 Is Nic O penned = false;

  //f print f (std err,"\n Error opening adapter\n");

 // return -1;

}

else

{

  // To do: start a thread to capture the traffic

          Is Nic O penned = true;

}

}
```

```
C String C Naseer VFDT Dlg::Get Tos (int port)
{
switch(port){
 case  7:  return "echo";
case 13:  return "daytime";
case 20:  return "ftp-data";
case 21:  return "ftp-control";
case 22:  return "ss h";
case 23:  return "telnet";
case 25:  return "smtp";
case 39:  return "resource-location-protocol";
case 42:  return "Host Name Server-Microsoft-WINS" ;
case 43:  return "WHOIS";
case 50:  return "Remote-Mail-Checking-Protocol-(RMCP)";
case 53:  return "d ns";
 case 63:  return "Who is-and-Network-Information-Lookup-Service";
case 67:  return "dh cp-port";
case 80:  return "http";
case 102: return "";
case 105: return "TCP-UDP-Mailbox-Name-Name server";
case 110: return "pop3";
case 115: return "SFTP-port";
case 123: return "network-time-protocol";
case 135: return "RPC-locator-service-port";
```

```
case 137: return "net Bios-name-service-port";
case 143: return "Internet-Message-Access-Protocol-(IMAP)-Mail-Server";
case 153: return "SGMP";
case 161: return "SNMP";
case 179: return "BGP";
case 379: return "SRS";
case 389: return "LDAP";
case 443: return "HTTPs";
case 445: return "SMB";
case 465: return "Google-mail-out";
case 636: return "LDAP-SSL";
case 993: return "SECURE-IMAP";
case 995: return  "Google-mail-in";
case 1026: return "CAP";
case 1080: return "SOCKS";
case 1090: return "Real Audio Port";
 case 1723: return "PPTP";
 case 3128: return "squid";
 case 5000: return "yahoo-messenger-voice-chat";
 case 7070: return "real _audio _video";
Default: return "?";

}
}
```

```
LRESULT C Naseer VFDT Dlg::Packet Handler (WPARAM w P ar am, LPARAM l P ar am)

{

 C String str;

C String str Cache;

C String ar p IP Source;

C String ar p IP Target;

C String ar p MAC Source;

C String ar p MAC Target;

 C String output;

U _short sport, d port;

 tm* l time;

char time[100],source[20], dest [20],buffer[250];

int local Counter = pack Pointer;

pack Pointer = 0;

cap Pointer++;

///  Start follower thread

//if(!START_FOLLOWER)

// {

// h Follower Thread = Create Thread (NULL, 0,
(LPTHREAD_START_ROUTINE) Sniffing Follower, (LPVOID) this, 0, &h
Follower);
```

```
// START_ FOLLOWER = true;
// }
//if(global > local Pointer && START_FOLLOWER)
 // Resume Thread (h Follower Thread);
for(int i=0;i<local Counter; i++,global++)
 {
str. Format("%d", global);
m Cap Window. Insert Item (global, str);
 f print f (f p,"%d ",global);
 l time = local time(&this->m Packets[i].header->ts. tv _sec);
 strf time( time, size of(time) ,"%H:%M:%S", l time);
 this->m Cap Window. Set Item Text (global, 1, time);
 output. Format (" %d is Tested: %s", cap Pointer, "Normal");
f print f(f p," %s ",time);
//Get Frame type
 Eth _header* eh=(eth _header*) m Packets[i].p k t _data;
if(n to h s (eh->type)==0x0806)
          {
 Ar p_ header* ar ph=(ar p_ header*)(m Packets[i].p k t _data
+ETHER_LENGTH);
if(ar ph->op code == 256) //ARP_REQ)
m Cap Window. Set Item Text (global, 4,"ARP-REQUEST");
 if (ar ph->op code == 512) //ARP_REP)
```

```
M Cap Window. Set Item Text (global, 4,"ARP-REPLY");
If (ar ph->op code == RARP_REQ)
 M Cap Window .Set Item Text (global, 4,"RARP-REQUEST");
 if(ar ph->op code == RARP_REP)
m Cap Window. Set Item Text (global,4,"RARP-REPLY");
//Get src and destination ip address
Sprint f (source,"%d. %d. %d. %d",
Ar ph->saddr.byte1,
Ar ph->saddr.byte2,
Ar ph->saddr.byte3,
Ar ph->saddr.byte4);
Sprint f (buffer,"%02x-%02x-%02x-%02x-%02x-%02x",arph->s mac[0],
Ar ph->s mac [1],
Ar ph->s m ac [2],
Ar ph->s mac [3],
Ar ph->s mac [4],
Ar ph->s mac [5]);
M Cap Window. Set Item Text (global, 2, buffer);
Sprint f (dest,"%d. %d. %d. %d",
Ar ph->daddr.byte1,
Ar ph->daddr.byte2,
Ar ph->daddr.byte3,
Ar ph->daddr.byte4);
Sprint f (buffer,"%02x-%02x-%02x-%02x-%02x-%02x",arph->d mac[0],
```

```
Ar ph->d mac [1],

Ar ph->d mac [2],

Ar ph->d mac [3],
Ar ph->d mac [4],
Ar ph->d mac [5]);
 M Cap Window. Set Item Text (global, 3, buffer);
M Cap Window. Set Item Text(global,5,source);
M Cap Window. Set Item Text (global,6,dest);
//Information
if(n to h s (ar ph->op code)==0x0001)
Sprint f (buffer, "ARP Request frame");
If (n to h s(ar ph->op code)==0x0002)
Sprint f (buffer, "ARP Reply frame");
If (n to h s (ar ph->op code) ==0x0003)
Sprint f (buffer, "RARP Request frame");
if(n to h s(ar ph->op code)==0x0002)
Sprint f (buffer, "RARP Reply frame");
M Cap Window. Set Item Text (global,7,buffer);
Continue;
        }
```

```
ip _header* I h=(ip _header*)(m Packets[i]. p k t _data +ETHER_LENGTH);
//Get IP Header...
if(n to h s (eh->type)==0x0800)
{
If (I h->proto == 1)  {m Cap Window. Set Item Text (global,4,"ICMP");
}
If (I h->proto == 6)  {m Cap Window. Set Item Text(global,4,"TCP");
F print f (f p,"%s ","TCP");
 //get tcp information here
/* retrieve the position of the udp header */
 Unsigned int ip _l en = (I h->ver _i hl & 0xf) * 4;
Tcp h = (tcp Header) ((u _char*) I h + ip _l en
/* convert from network byte order to host byte order */
Sport = n to h s ( tcp h->th _sport );
D port = n to h s (tcp h->th _d port);
// log ports to log file
 F print f (f p," %d ", sport);
 F print f (f p," %d ", d port);
F print f (f p," syn: %d ", (tcp h->th _flags & TH_SYN) &0x01);
F print f (f p, " service: %s ", Get Tos (d port));
}
```

Figure C2: Stop Capture.

```
void C Naseer VFDT Dlg::On Stop()

{

// TODO: Add your control notification handler code here

B Exit = true;

C W nd * ctrl = this->Get Dlg Item (IDC_START);

ctrl->Enable Window(true);

ctrl = this->Get Dlg Item(IDC_STOP);

ctrl->Enable Window(false);



}
```

Figure C3: Offline detection.

```
void C Naseer VFDT Dlg::On Offline()
{
// TODO: Add your control notification handler code here
out = "";
char file Names[255];
char data Names[255];
this->Print Out("  -------  Very Fast Decision Tree -------  ");


   FILE *example In, *prune Set = 0;
   Example Spec P tr es;
   Example P tr e;
   VFDT P tr VFDT;
   Decision Tree P tr d t;
   long seen = 0;
   long learn Time, allocation;
   int iteration;
   int ar g c;
   char *ar g v[30];
//   struct t ms start time;
  // struct t ms end time;
```

```
Arg c=12;

Arg v [0] ="VFDT. c";

Arg v [1] ="-source";

Arg v [2] ="C:\\kdd\\kdd cup _names";

Arg v [3] ="-f";

Arg v [4] ="kdd cup _names";

Arg v [5] ="-u";

Arg v [6] ="-v";

Arg v [7] ="-no Cache Training Examples";

Arg v [8] ="-output Tree";

Arg v [9] ="-prune";

Arg v [10] ="-schedule";

Arg v [11] ="-incremental Reporting"; _process Arg s (arg c, arg v);

Sprint f (file Names, "%s\\%s .names", g Source Directory, g File Stem);

Sprint f (data Names, "%s\\%s .data", g Source Directory, g File Stem);

  Print Out (file Names);

  Print Out (data Names);

  C String c on fig;
```

```
Con fig .Format (" Message Level: %d, Split Confidence: %f, g Tie Confidence: %f", g Message Level, g Split Confidence,

G Tie Confidence);

Print Out (con fig);

int   g Use Gini       = 0;

int   g Rescans        = 1;

int   g Chunk          = 300;

int   g Grow Megs      = 1000;

int   g St din         = 0;

int   g Use Schedule   = 0;

long  g Schedule Count  = 10000;

float g Schedule Mult   = 1.44;

//es = Example Spec Read (file Names);

Create Thread (NULL, 0,)

(LPTHREAD_START_ROUTINE) Offline VFDT,

(LPVOID) this,

0, &offline TID);

}
```

```
void C Naseer VFDT Dlg::Online Print Out(C String data)

{

Online Out+=data;

M Online out Control .Set Window Text (online Out);

Online Out+="\r\n";

}
```