Anomaly Based Network Intrusion Detection Using Data Mining Techniques

A thesis submitted in part fulfillment of the requirements for award of the degree of Doctor of Philosophy

Prasanta Gogoi
Registration No. 054/1999

School of Engineering
Department of Computer Science and Engineering
Tezpur University

November 2012

Dedicated

To my

Parents and Family

Abstract

With rapid growth of high speed networks, faster computations and efficient high volume data transfers, society's dependence on computers, networks and the Internet has risen tremendously. At the current high level of computer utilization, system-generated or artificially generated threats to our computer infrastructure have also risen immensely. Many new and sophisticated protective measures have been developed to counter threats. These measures become obsolete after a short duration of time as they fight against highly intelligent software generated threats. A threat to a computer network is considered an intrusion or attack. Network intrusion detection is carried out in two ways: misuse or signature based and anomaly based. A misuse based network intrusion detection system (NIDS) performs detection on the basis of prior attack instances. Thus, this method of detection cannot detect new or unknown attacks. On the other hand, an anomaly based network intrusion detection system (ANIDS) attempts to detect new or unknown attacks from the current attack instances, using innovative attack detection methods. In current research in the domain of network intrusion detection, anomaly based network intrusion detection has drawn in-depth attention because of its inherent ability to detect unknown attacks without needing instances of prior attack data. Anomaly based network intrusion detection is closely related to the analysis of network traffic data. Network traffic data is high dimensional and occurs in high volumes with numerical, categorical and mixed attributes. Data mining techniques are suitable for analyzing large volume data to discover nontrivial embedded patterns (regularities and relationships). Data mining has the potential to provide efficient and effective solutions to the problem of network intrusion detection where handling huge volumes of high dimensional data is an absolute necessity.

In this thesis, we provide an in-depth study of weaknesses in networks from the view point of attackers, types of network intrusions, and network intrusion detection methods using data mining techniques. We discuss fundamentals of computer networks, intrusion detection methods, characteristics of intrusion data, generation of intrusion data, and evaluation metrics for detection methods. Apart from two exhaustive surveys on network anomaly detection methods and outlier based anomaly detection methods, our main contributions are four anomaly based intrusion detection methods included in this thesis.

- A supervised learning method for anomaly based intrusion detection, which builds a model on the basis of pre-labeled training data.

- An unsupervised learning method for anomaly based intrusion detection which can detect known as well as unknown attacks using analysis and extraction of interesting relationships from input data.

- An outlier based network anomaly detection method which can identify rare class attacks by isolating or partitioning the rarely occurring instances or instances that are dissimilar from the majority of instances.

- A multi-level hybrid anomaly based network intrusion detection method where competing supervised and unsupervised intrusion detection methods with their inherent advantages are combined appropriately for the purpose of network intrusion detection with high detection accuracy.

Experimental results on benchmark public and real life private datasets establish the validity of the proposed methods.

Keywords: *Network, intrusion, masquerade, misuse, anomaly, supervised, unsupervised, outlier, multi-level, hybrid, packet, netflow, DoS, vulnerability, signature, recall, precision, true positive, false positive.*

Declaration

I, Prasanta Gogoi, declare that the thesis entitled "Anomaly Based Network Intrusion Detection Using Data Mining Techniques" and the work presented in it is my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at Tezpur University.
- where any part of this thesis has not previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- where I have consulted the published work of others, this is always clearly attributed.
- where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.

Signed: Prasanta Gogoi

Date: 12/04/2013

Place: Tezpur, INDIA

School of Engineering
Department of Computer Science and Engineering
Tezpur University

Dr. Dhruba Kumar	Phone: 03712 - 267007
Bhattacharyya	Fax: 03712 - 267005
Professor	e-mail: dkb@tezu.ernet.in

Certificate

This is to certify that the thesis entitled "Anomaly Based Network Intrusion Detection Using Data Mining Techniques" submitted to the School of Engineering of Tezpur University in part fulfillment of the requirements for the award of the degree of Doctor of Philosophy in Computer Science and Engineering is a record of research work carried out by Mr. Prasanta Gogoi under my supervision and guidance.

All helps received by him from various sources have been duly acknowledged.

No part of this thesis has been reproduced elsewhere for award of any other degree.

12/4/13

(Dhruba Kumar Bhattacharyya)

School of Engineering
Department of Computer Science and Engineering
Tezpur University

Dr. Bhogeswar Borah
Associate Professor

Phone: 03712 - 267009
Fax: 03712 - 267005
e-mail: bgb@tezu.ernet.in

Certificate

This is to certify that the thesis entitled "Anomaly Based Network Intrusion Detection Using Data Mining Techniques" submitted to the School of Engineering of Tezpur University in part fulfillment of the requirements for the award of the degree of Doctor of Philosophy in Computer Science and Engineering is a record of research work carried out by Mr. Prasanta Gogoi under my supervision and guidance.

All helps received by him from various sources have been duly acknowledged.

No part of this thesis has been reproduced elsewhere for award of any other degree.

Borah
12/4/2013
(Bhogeswar Borah)

Preface

The present PhD thesis has been accomplished at the Tezpur University, Tezpur, India. The supervision was provided by Dr. Dhruba Kumar Bhattacharyya, and Dr. Bhogeswar Borah, Department of Computer Science and Engineering, Tezpur University.

All experiments were conducted at the Network Security Laboratory, Department of Computer Science and Engineering, School of Engineering, Tezpur University. The studies presented in the thesis were approved by the ethical committee.

Tezpur, November, 2012

Prasanta Gogoi

Thesis submitted: November, 2012

Thesis defended:

Acknowledgement

I am indebted to my supervisor Prof. Dhruba Kumar Bhattacharyya for getting me involved in this research in the first place; and very thankful for his guidance and support through the years. I appreciate his considerate, encouraging, and understanding personality as a mentor, who set realistic goals that always kept me motivated for this research. He always gave positive feedback and had a good sense of humor while dealing with disappointments or unforeseen circumstances. I am thankful for his encouragement and help which enabled me to complete this dissertation.

I am indebted to my co-supervisor Dr. Bhogeswar Borah for his guidance and support through the years.

I am indebted to Prof. Jugal K Kalita of University of Colorado at Colorado Springs, USA for his constant guidance and support.

I am also grateful to the members of the doctoral committee, Prof. D. K. Saikia and Dr. N. Sharma. As individuals, they offered encouragement and constructive criticism for this dissertation to take its current shape.

I would like to thank my lab mates for their friendship and for sharing days and nights with me in the research lab and all the members of the Network Security Laboratory, Tezpur University who gave me the possibility to complete. I am grateful to my parents, brothers, sisters and family members for their support throughout my long education life.

Tezpur University

Certificate

This is to certify that the thesis entitled "Anomaly Based Network Intrusion Detection Using Data Mining Techniques" submitted by Mr. Prasanta Gogoi to the School of Engineering of Tezpur University in part fulfillment of the requirements for the award of the degree of Doctor of Philosophy in Computer Science and Engineering has been examined by us on ______ and found to be satisfactory.

The committee recommends for award of the degree of Doctor of Philosophy.

Signature of

Supervisor External Examiner

Date:

Contents

List of Tables

List of Figures

List of Algorithms

List of Abbreviations

ADAM	Audit Data Analysis and Mining
ADSL	Asymmetric Digital Subscriber Line
ADWICE	Anomaly Detection With fast Incremental Clustering
ANIDS	Anomaly based Network Intrusion Detection System
ARM	Association Rule Mining
bps	bits per second
DARPA	Defense Advanced Research Projects Agency
DDoS	Distributed Denial of Service
DMZ	Demilitarized Zone
DoS	Denial of Service
EM	Expectation Maximization
EMERALD	Event Monitoring Enabling Responses to Anomalous Live Disturbances
FIRE	Fuzzy Intrusion Recognition Engine
FN	False Negative
FNNk	Forward Nearest Neighbour Set of k objects
FNOFk	Forward Neighbour Outlier Factor of k objects
FP	False Positive
fpMAFIA	Frequent Pattern Merging of Adaptive Finite Intervals Algorithm
FTP	File Transfer Protocol
GMM	Gaussian Mixture Model
HMM	Hidden Markov Models
HTTP	Hyper Text Transport Protocol

ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
ISO	International Organization for Standardization
ISO/OSI	International Organization for Standardization/ Open Systems Interconnection
ISP	Internet Service Provider
KDD	Knowledge Data Discovery
KNN	*k* Nearest Neighbour
LAN	Local Area Networks
LOF	Local Outlier Factor
MAN	Metropolitan Area Network
MINDS	Minnesota Intrusion Detection System
MLH-IDS	Multi-level Hybrid Intrusion Detection System
MLP	Multi-layered Perception
NN	Neural Networks
NNID	Neural Network Intrusion Detector
NNk	Nearest Neighbour Set of *k* objects
OCSVM	One Class Support Vector Machine
OSI	Open Systems Interconnection
PCC	Percentage of Correct Classification
PDU	Protocol Data Unit
PPP	Point-to-Point Protocol
R2L	Remote to Local

ROC	Receiver Operating Characteristics
RST	Rough Set Theory
RT-UNNID	Real Time Unsupervised Neural Network Intrusion Detector
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
SVM	Support Vector Machine
TCP	Transport Control Protocol
TCP/IP	Transport Control Protocol/ Internet Protocol
TN	True Negative
TP	True Positive
TPDU	Transport Protocol Data Unit
TRILL	TRansparent Interconnection of Lots of Links
TUIDS	Tezpur University Intrusion Detection System
U2R	User to Root
UDP	User Datagram Proctocol
UNADA	Unsupervised Network Anomaly Detection Algorithm
USB	Universal Serial Bus
WAN	Wide Area Networks

List of Symbols

A_1	Attribute	146
A_2	Attribute	146
A_d	Attribute	146
D_1	Domain	146
D_2	Domain	146
D_d	Domain	146
X	Object	146
C	Cluster	147
x_1	Object	146
x_2	Object	146
x_d	Object	146
N	No. of objects	146
M	No. of attributes	146
$MinAtt$	Threshold value for minimum no. of attributes	147
$minSize$	Threshold value for minimum objects in a cluster	150
$sim(X, Y)$	Similarity value between two objects	145
$sim(C, X)$	Similarity value between a cluster and an object	147
$s(x_j, y_j)$	Similarity for j^{th} attribute between two objects	145
$s(v_j, x_{a_j})$	Similarity for j^{th} attribute between cluster and object	146
$Olist$	Object list in a cluster	147
$Profile$	Profile of a cluster	147
$noAttributes$	Total number of attributes in a cluster profile	147
$Dattributes$	Distribution of attributes in a cluster profile	147
$Values$	Values of attributes in a cluster profile	147
X_i	Object	148
SC	Supervised cluster	149
UC	Unsupervised cluster	149
MIN	No. of attributes	150
$MinmAtt$	Threshold value for minimum no. of attributes	194
D_{attrib}	Distribution of attributes in a cluster profile	194
N_{attrib}	Total number of attributes in a cluster profile	194
V_{attrib}	Values of attributes in a cluster profile	194
$Clusprofile$	Profile of a cluster	194
$Oblist$	Object list in a cluster	194

List of Symbols

$simO(X, Y)$	Similarity value between two objects	193
$simC(C, Y)$	Similarity value between a cluster and an object	194
$\|D\|$	Size of dataset D	214
$dist(p, q)$	Distance between objects p and q	214
$NNk(p)$	Nearest neighbor set of k objects for object p	214
$FNNk(p)$	Forward nearest neighbor set of k objects for object p	214
$FNOFk(p)$	Forward neighbor outlier factor for object p	214
M_{th}	Threshold value for forward neighbor outlier factor	216
$getNNk()$	Function for computing nearest neighbor set of k objects	218
$getFNOFk()$	Function for computing forward neighbor outlier factor	218
C_1	Classifier	242
C_2	Classifier	242
C_N	Classifier	242
S_1	Sample dataset	242
S_2	Sample dataset	242
S_M	Sample dataset	242
S_i	Sample dataset	242
t_i	Test instance	242
D	Dataset	242

CHAPTER 1

Inroduction

Contents

1.1 Introduction

Network based computer systems play vital role in modern society for information. The dependency on network based computer system is inevitable for all financial and other transactions, business processes, government administrations and social communications. These interconnected computer systems are transforming our day-to-day activities like financial or government administrations or other different transactions very fast and easily accessible. The Internet is a broader network of interconnected computer systems. It is the network of networks. The role of the Internet is to convey information of all types, starting from simple binary data transfer to complex real-time multimedia data transmission. With the evolutionary expansion of computer networks and its increased usage, they have become the target of enemies and criminals. The security of computer system is compromised when network intrusion occurs. The intrusions or attacks to the computer or network systems are the activity or attempt to destabilize it by compromising the computer security in confidentiality, availability or integrity of the system.

1.1.1 Recent Network and The Internet

Networks are created with several individual entities to perform complex interactions. It provides people and machine varying communication services. A computer network is comprised of several physical and software components. Networking functions are designed as a layered model viz., ISO/OSI (International Organization for Standardization/ Open Systems Interconnection), TCP/IP (Transport Control Protocol/ Internet Protocol). Each layer is responsible for a different facet of communications independently without affecting the adjacent layer. Each individual entity of a network layer is assigned a definite role to play towards the ensemble behaviour of the network. Networks now have to confront an unprecedented range of threats, specifically, attacks and vulnerabilities. Figure 1 1 shows continuous growth of the Internet users[1]. However, with this increasing growth of the Internet users, the number and types of threats to disrupt the normal Internet activities also have been increasing significantly. A list of top 20 most commonly used mali-

[1] http://www.isc.org/solutions/survey/history

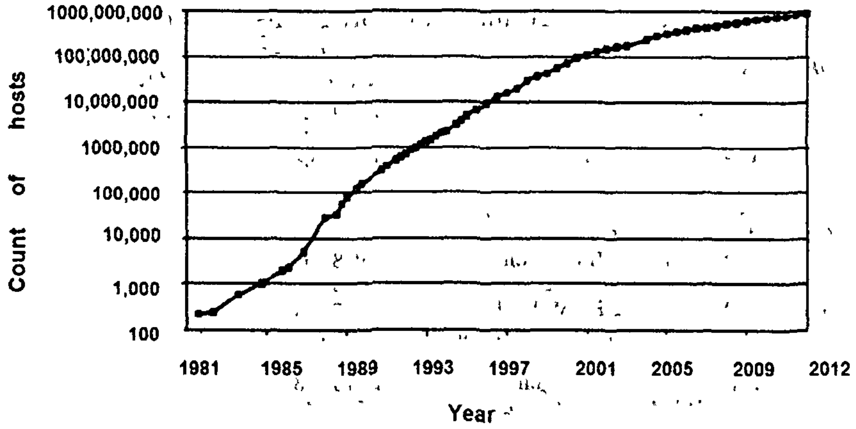


Figure 1.1: Internet host 1981-2012 Statistics

cious programs[2] involved in the Internet attacks in the year 2011 is shown in Table 1.1.

1.1.2 Network Vulnerabilities

Network vulnerabilities are inherent weaknesses in the design, configuration, or implementation of computer networks that render it susceptible to a threat of security. Threat may arise from exploitation of design flaws of hardware and software of computer network systems. Systems may be incorrectly configured, and therefore vulnerable to attack. The vulnerabilities of this kind generally occur from inexperience, insufficient training, or half done work. Another vulnerability source is the poor management such as inadequate procedures and insufficient checks of the network systems.

1.1.3 Anomalies in Networks

A computer network is formed based on the contributions of several individual entities towards providing several complex communication services. A network is made of several components. Networking functions are structured as a layered model viz., ISO/OSI, TCP/IP. Each layer is trusted for a different part

[2] http://www.securelist.com/Kaspersky/Security/Bulletin/2010/Statistics/2010

Table 1.1: The Top 20 malicious programs on the Internet

Rank	*Name*	*Number of Attacks*	*% of All Attacks*
1	Malicious URL	712,999,644	75.01%
2	Trojan.Script.Iframer	35,522.262	3.67%
3	Exploit.Script.Generic	17,176,066	1.81%
4	Trojan.Script.Generic	15,760,473	1.66%
5	Trojan-Downloader.Script.Generic	10,445,279	1.10%
6	Trojan.Win32.Generic	10,241,588	1.08%
7	AdWare.Win32.HotBar.dh	7,038,405	0.74%
8	Trojan.JS.Popupper.aw	5,128,483	0.54%
9	AdWare.Win32.FunWeb.kd	2,167,974	0.23%
10	Trojan-Downloader.Win32.Generic	1,979,322	0.21%
11	AdWare.Win32.Eorezo.heur	1,911,042	0.20%
12	AdWare.Win32.Zwangi.heur	1,676,633	0.18%
13	Hoax.Win32.ArchSMS.heur	1,596,642	0.17%
14	Trojan.HTML.Iframe.dl	1,593,268	0.17%
15	Trojan.JS.Agent.uo	1,338,965	0.14%
16	AdWare.Win32.FunWeb.jp	1,294,786	0.14%
17	Trojan-Ransom.Win32.Digitala.bpk	1,189,324	0.13%
18	Trojan.JS.Iframe.tm	1,048,962	0.11%
19	AdWare.Win32.Agent.uxx	992,971	.10%
20	AdWare.Win32.Shopper.ee	970,557	.10%

of communications independently and without affecting the adjacent layer. The individual entities participating in the formation of the network play a definite role towards a common ensemble behaviour of the network. The normal network behaviour thus obtained is then used for detection of network anomalies.

An anomaly is an occurrence which is suspicious from security perspective. Typically, network anomalies refer to consequent situation when network activities deviate significantly from normal behaviour of network activity. Network anomalies may arise due to a number of causes, such as network overload, malfunctioning of network devices and network attacks or intrusions which disrupt the delivery of normal network services. These anomaly events disrupt the normal network behaviour to some of measurable parameter in network data.

1.1.4 Detecting Network Anomalies using Data Mining

Anomalies are events deviating from the normal behaviour, and are suspected from the security perspective. Anomalies in a network may occur due to two major reasons[6]: performance related and security related. A performance related anomaly may occur due to several malfunctioning activities, such as:

network device failure (e.g., router misconfiguration), network overloading etc. However, security related anomalies occur due to intrusion activities in the network. The severity of the disruption of the normal behaviour largely depends on the type, duration and mode of occurrence of the security related anomalies. Such activities are caused by two basic types of users: inside users (compromised) and outside attacker. Such security related anomalies can be of six basic types[7] based on their characteristics or features, *Infection*, *Exploding*, *Probe*, *Cheat*, *Traverse* and *Concurrency*. However, in literature[8], anomalies are in general classified into three broad categories: point anomalies, contextual anomalies and collective anomalies. An anomaly based network intrusion detection system (ANIDS) attempts to detect the anomalies before they inflict the computer networks. Anomaly based detection needs multiple steps of operation for performing anomaly detection. Anomaly based network intrusion detection approach, typically, builds a model of normal system behaviour from the observed data and distinguishes any significant deviations or exceptions from this model. Anomaly based detection implicitly assumes that any deviation from normal behaviour is anomalous. This detection approach has the ability to examine and detect new or unknown attacks. A major issue in the anomaly based intrusion detection is the selection of appropriate threshold(s) for accurate identification of normal or anomalous activities. Anomaly based detection methods are also computationally expensive because of the overhead of keeping track of computation and possibly updating several profile matrices.

1.2 Data Mining

Rapid advances in data capture, transmission and storage technologies have enabled modern business and science to collect increasingly large volume of data. Retailers are accumulating their daily transactions into large databases. Besides direct use of the databases the enterprizes can benefit in immense from the volume of gathered past data. In the domain of scientific community, large amount of network data, remote sensing data, protein data and genome data are collected for inferring some valuable information from them. Data mining[9,10,11] is the technique of analyzing such voluminous data in order to extract valid, potentially useful and meaningful information that might otherwise remain unknown. It is defined in[11] as follows:

Data mining is the analysis of observational large dataset to find unsuspected relationships and summarize it in novel ways such that the data is understandable and useful to its owner

The relationship and summaries, for example, linear equations, rules, clusters, graphs etc., are usually referred to as patterns or models which are derived by a data mining exercise. A global summary of dataset is the model structure whereas statements of restricted regions of space covered by the variable are represented by a pattern structure. The observational data referred to in the definition deals with data which have been already collected with subjective of other purpose apart from analysis of data mining. Thus, the data collection strategy is not the objective of data mining process. For this reason data mining is often referred to as secondary data analysis.

1.2.1 Data Mining Tasks

Data mining is categorized into different types of tasks based upon the different types of models or patterns they find. In practice, data mining tasks are of two categories: predictive and descriptive. Predictive mining tasks make inference on analyzing the current data to make subsequent predictions. Descriptive mining tasks perform characterization of the properties of the data in the database. Some of the important data mining tasks[9] include:

1. *Cluster analysis*: Cluster analysis performs grouping of data objects on the basis of information extracted from the data which describes the data and their relationships. The goal of cluster analysis is to partition a set of data objects into distinct groups. In the groups, similar characteristic objects are gathered together and dissimilar characteristic objects are collected in different groups. The distinct and good clustering occurs by the way of maximum similarity within a group and maximum difference among the groups.

2. *Association Rule Mining*: Frequent patterns are those which occur frequently in data. Frequent pattern mining leads to the extraction of correlations and interesting associations within data.

3. *Outlier mining*: A dataset may contain data objects whose characteristics are significantly different from the rest of the data. The significantly different data objects in a dataset are known as outliers. Some particular applications, for example, credit card fraud detection, rare occurring events are more interesting compared to the regularly and normal occurring ones. The outlier mining refers to analysis of outlier data in a dataset.

4. *Classification and prediction*: Classification is a process of finding a model (or function), which describes and recognizes differences in data label or classes, for the purpose of using the model to predict the label or class of objects of unknown class label. The model is derived from the analysis of training data whose instances have known class labels.

5. *Evolution analysis*: Evolution analysis in data describes and builds model or finds regularities for objects which exhibit changes of behaviour over time.

Our work is primarily related to network anomaly detection which is more connected to detection of attacks of known, unknown as well as rare categories. Therefore, we concentrate specifically on clustering, classification and outlier mining.

1.3 Existing Network Anomaly Detection Methods

Over the past couple of years, a large number of network anomaly detection methods have been developed. Most of these methods have their limitations in size of data input, real time attack detection, all the known as well as novel attack detection and performance of detection rate. However, the existing network anomaly detection methods can broadly be classified into the following categories.

1.3.1 Supervised Learning

In supervised learning approach[12,13,14,15,16] of network anomaly detection, a model is developed based on labeled training data. Anomaly detection builds models for normal behaviour and significant deviations from it are flagged as attack. The supervised classification method uses the training algorithm to create a set of representative classes from the available labeled training objects. Unlabeled test objects are then inserted in these representative classes based on similarity estimation using similarity measures and thus get labels of the classes in which they are inserted. Classes are described by object list belonging to classes and the profile of parameters representing the class.

1.3.2 Unsupervised Learning

The objective of a network based intrusion detection is to handle accurately the threats that arise from new or previously not known intrusions. The possible detection approach of novel intrusions is anomaly based detection approach instead of the rule based approach. In anomaly based supervised detection approach, obtaining labeled or purely normal data is a critical issue. Whereas, unsupervised anomaly based detection[17,18,19,20,21,22,23] can address this issue of novel intrusion detection without prior knowledge of intrusions or purely normal data. Clustering is a widely found method for anomaly based unsupervised detection of intrusions. In order to label clusters, unsupervised anomaly based detection approach uses two assumptions: (i) the number of normal instances vastly outnumber the number of anomalies, and (ii) anomalies themselves are qualitatively different from the normal instances.

1.3.3 Probabilistic Learning

A probabilistic learning mechanism enables us to evaluate a system's performance based on a probabilistic uncertainty or by randomness. The important characteristic of probabilistic learning is its capability to update prior outcome of evaluations by reforming them with results of recent obtained information. Some of the popular methods of this category are: Bayesian network[24,25,26], Naive Bayes[25,27], Hidden Markov Model[28,29], Gaussian Mixture Model[30,31] and Expectation Maximization Method[32].

1.3.4 Soft Computing

Soft Computing aims to provide realistic solutions for any computationally intelligent problem based on individual or a combination of several emerging problem solving technologies, such as Fuzzy Logic[33], Probabilistic Reasoning[34], Neural Networks[35], and Genetic Algorithms[36]. For solving complex and real-world problems, these soft computing methods provide new complementary reasoning and searching techniques. In comparison to procedural ordinary hard computing, soft computing is more tolerant of imprecision, partial truth and uncertainty[37]. Soft computing based anomaly detection include approaches of ANN[38], Rough Set[39], Fuzzy Logic[33] and Evolutionary Computing[40].

1.3.5 Knowledge Based

A knowledge based system is a problem solving and decision making system based on knowledge of its task and logical rules or procedures for using knowledge. Knowledge acquisition, knowledge representation, and application of large bodies of knowledge to the particular problem domain are the key factors of knowledge based methods. Expert systems[41,42] is a commonly found knowledge based method.

1.3.6 Hybrid Learning

The problem of detecting new attacks poses a special challenge in network intrusion detection. Within such a context, the design approach to the creation of an intrusion detection system resting on the multi-level classifiers has brought many remarkable contributions to the intrusion detection domain. The performance of a single classifier is not equally good for classification of all categories of attack as well as normal instances. There is a possibility of obtaining good classification accuracy for all categories in a dataset by using an appropriate combination of multiple well performing classifiers. A multi-level hybrid intrusion detection system (MLH-IDS) is the combination of different well performing methods viz., supervised, unsupervised, outlier based method. This approach works in multiple steps of operations on unclassified data. The prime issues of the approach are selection of methods and combination of methods for the desired anomaly detection. Some of the hybrid supervised and unsupervised methods can be found in[43,44,45,46,47]

1.4 Motivation

The computer network is a complex and highly structured system. It is an ensemble of different softwares and hardware devices. The presence of anomalies causes degradation of performance and security related problems in a network. To counter these security related problems, in the past couple of years several novel and significant solutions have been evolved. However, with the increased sophistication in attackers approach, and with the increasingly fast network transmission technology, most existing methods have been found incapable of handling the recent attacks with high accuracy. Thus the need to combat the newer computer and network attacks is becoming increasingly important In view of these, our focus of research is the network anomaly detection with best possible accuracy.

Our motivation is to build efficient and effective network anomaly detection methods using data mining techniques for detection of known as well unknown attacks with higher detection accuracy and lower false detection rate compared to existing anomaly detection methods

1.5 Research Issues

Signature based or misuse based schemes of intrusion detection provide an effective detection rate for specified and well-known attacks. However, their ability is limited to only well-known and familiar intrusions. With minimum variation(s) of those already known attacks[48], such detection schemes often can be found ineffective in terms of detection accuracy. On the contrary, an anomaly based intrusion detection scheme is built mainly to detect new or previously unknown events, apart from the detection of known attacks.

With the increasing sophistication and skilled involvement of professionals in the intrusion activities, this approach for detection of novel intrusions has been receiving significant interest among the network security researchers. Various network anomaly detection methods are already in existence, and many schemes with new techniques are being introduced. Although, the subject is yet far from maturity and vital key objectives remain to be unsolved. Building a robust anomaly detection method which can detect all normal activities as well as massive or stand alone attacks faster is yet a research objective for researchers in this area. Some of the important issues are identified as:

- Determination of an appropriate proximity measure for cluster expansion and profile generation from intrusion data.
- Identify relevant features of intrusions for normal as well as all known classes of attacks.
- Development of faster/near real time method for detection of commonly occurring known as well as unknown attacks.
- Development of data mining method for detection of known as well as unknown rare class attacks.
- Development of an ensemble of supervised and unsupervised method for detection of known and unknown attacks with minimum false alarms.

1.6 Contributions

In this thesis work, we summarize fundamentals of computer networks, sources of anomalies of network services, detection methods for network anomalies, description for required and acquired datasets, structured classification of existing network anomaly detection methods, and we develop methods for network anomaly detection using various data mining approaches and their evaluations. Our contributions are enlisted as follows:

1. A concise description of fundamentals of computer networks and its necessary softwares and hardwares, sources of network anomalies and various detection methods with inclusion of relevant features and acquisition of datasets and its performance evaluation metrics.
2. A summarization of existing network anomaly detection methods under classification of six broad categories and their brief descriptions.

3. A supervised anomaly detection method on the basis of incremental clustering algorithm with two new similarity measures for generation of profiles for normal and attack categories. The method is evaluated with excellent results using benchmark and real life datasets.

4. An unsupervised clustering algorithm, *k-point*, is developed for unknown anomaly detection based on two similarity measures. The method is evaluated for satisfactory performance results using popular benchmark and captured real time intrusion datasets.

5. Two similarity measures for mixed type data, both categorical and numerical, are developed which are used in clustering and profile generation in supervised anomaly detection method. Similarly, another two similarity measures are developed and used for clustering in the unsupervised method of anomaly detection.

6. An outlier detection algorithm is developed highlighting an outlier factor based on symmetric neighbourhood relationship for network anomaly detection. The method is tested for satisfactory performance in few category of network attacks using popular benchmark and captured real time intrusion datasets.

7. A hybrid multi-level intrusion detection method is developed combining classifiers from supervised, unsupervised and outlier detection methods. The method is evaluated for excellent performance using popular benchmark and captured real time intrusion datasets.

1.7 Chapter Organization

The thesis work is elaborated in nine broad chapters including this *Chapter of Introduction* about computer network, its anomalies and detection methods. The organization of the rest of the thesis is prepared as follows:

- *Chapter 2* provides an elaborate description about fundamentals of computer and networks, software and hardware required in computer networks and sources and generation of network anomalies and their effect on network.

1.7. Chapter Organization

- *Chapter 3* summarizes the network anomaly detection methods and their issues, description of popular benchmark and captured real time network intrusion datasets and description of performance evaluation metrics for anomaly detection methods.

- *Chapter 4* presents a comprehensive and structured survey on existing methods and systems for network anomaly detection. This chapter draws on techniques from many diverse areas of intrusion detection. The methods are classified into six broad categories.

- *Chapter 5* presents a supervised anomaly detection methods on the basis of incremental clustering algorithm using two new similarity measures and evaluation of the method with the benchmark and real time datasets.

- *Chapter 6* describes an unsupervised anomaly detection method on the basis of developed clustering algorithm *k-point* and similarity measures, and evaluation of the method with benchmark and real time intrusion datasets.

- *Chapter 7* presents a outlier mining method for anomaly detection on the basis of developed outlier detection algorithm using symmetric neighbourhood relationship and outlier factor and evaluation of the method.

- *Chapter 8* provides a multi-level hybrid intrusion detection method that uses a combination of supervised, unsupervised and outlier based methods to achieve the best detection performances for both known and unknown attacks in network anomaly detection.

- *Chapter 9* finally, concludes the thesis work and gives possible directions for future research in network anomaly detection.

CHAPTER 2

Background

Contents

The contents of this chapter is organized into two distinct parts to provide a background of networks and their anomalies. *Part I* of this chapter shall provide a background on basics of networking, its components, types, scales, topologies and various performance constraints. *Part II* is dedicated in introducing the concept of network anomalies, their causes; sources and precursors. It will also attempt to highlight the various classes of network intrusions or attacks.

2.1 Part I: Basics of a Network

Networks are complex interacting systems and are comprised of many individual entities. Two or more computer systems capable of sending or receiving data from each other through a shared-access medium, are said to be connected. The behaviour of the individual entities contributes to the ensemble behaviour of the network. In a computer network, there are generally three communicating entities: (a) *Users*: human entities responsible for various actions in the network, (b) *Hosts*: an entity which can be identified with a unique address, and (c) *Processes*: instances of executable programs used in a client-server based architecture. Client processes request the server(s) for a network service, whereas the server processes provide the requested services to the clients.

2.1.1 Typical View of a Network

People usually think of a computer network as the sum of connections among computers to allow communication among computer systems or devices. In this simple view of a network, a set of computers or devices is shown as connected to each other with the ability of exchanging data. Figure 2.1 shows a typical view of computer networking. It can be seen from Figure 2.1 that computer nodes or servers or devices can be connected through some special devices, like switch, router or a hub by using appropriate communication media. Apart from the softwares used to support communication, the performance of data exchange among these nodes, servers or devices largely depends on the type of media and the connecting device used.

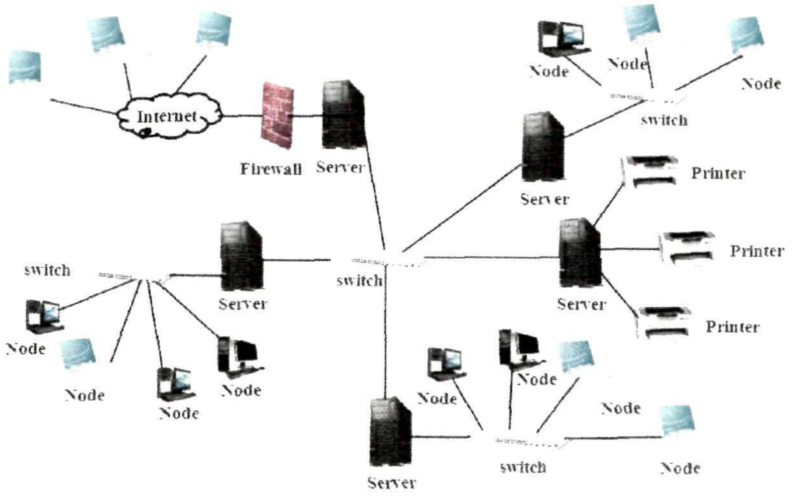


Figure 2.1: A typical computer networking

2.1.2 Communication Media

In the past couple of years communication media technology has achieved a remarkable progress in terms of speed, reliability and robustness. Basically, the function of communication media is to transport raw bit streams from one computer to another. In this section we highlight the existing developments in communication media. We will have several options for selection of the appropriate communication medium for the actual transmission. Apart from highlighting these media, we shall also analyze the performance of these media types in terms of four crucial parameters: (i) bandwidth, (ii) delay, (iii) cost, and (iv) installation and maintenance. Here, for the sake of understanding, we discuss the existing media types in two major categories [49,50]: (i) *Guided media* and (ii) *Unguided media*. Figure 2.2 shows a basic classification of the existing popular communication media. A general performance comparison between the two communication media also has been reported in Table 2.1. It can be easily seen from the table that both media types have their own merits and demerits.

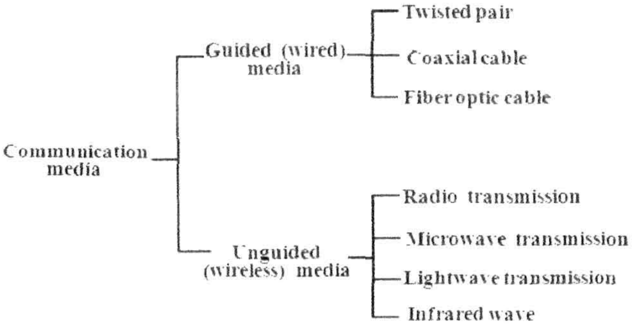


Figure 2.2: Classification of Communication Media

Table 2.1: Performance Comparison of Guided and Unguided Media

	Guided Media	Unguided Media
1	Contains the signal energy within a solid medium and propagates in guided form.	Signal energy is not contained and propagates in unguided form as electromagnetic waves.
2	Point to point communication.	Radio broadcasting in all directions.
3	Discrete network topologies are relevant.	Continuous network topologies are relevant.
4	Attenuation depends on the distance exponentially.	Attenuation is proportional to square of the distance.
5	Scopes for enhancing the transmission capacity by increasing number of cables.	No scopes for enhancement.
6	Installation may be costly and time consuming.	Relatively less costly and less time consuming.

2.1.2.1 Guided Media

As reported in Table 2.1, in this type of media, signal energy is contained and guided within a solid media and communication type is point-to-point. The three popular instances of this category of media: (i) Twisted pair, (ii) Coaxial cable and (iii) Fibre optic cables, are described in brief next.

Twisted Pair

This type of medium is built by using two insulated copper wires (1 mm thick) twisted together like a DNA molecule, i.e., in a helical form. It uses the wires in a twisted form because the waves from different twists cancel out, so the radiation from the wires is less. The bandwidth of a twisted pair is dependent on two important factors: (i) the wire thickness and (ii) the traveled distance. Generally, in most cases, it allows several megabits/sec to travel for a few kilometers. The two major causes behind the popularity of twisted pair cable are: (i) Adequate performance and (ii) Low cost.

Coaxial Cable

Coaxial cables are popular for its well structure and shielding mechanism, which enables it to provide a good combination of high bandwidth and excellent noise immunity. It is made of a stiff copper wire as the core, surrounded by an insulating material, encased by a cylindrical conductor, i.e., a closely-woven braided mesh. A protective plastic sheath covers the outer conductor. Today, a coaxial cable has a bandwidth of approximately 1 GHz. This medium is widely used for cable television and metropolitan area networks.

Fiber Optic Cables

This media type is similar to coaxial cable, except without the braid. Figure 2.3 (a) shows a single fiber optic cable, where the center is the glass core which allows to propagate light. The diameter of the core varies based on the type of fibre used. In multimode fibers, it is typically 50 *microns* in diameter, whereas, in the single-mode fibers, it is 8-10 *microns*. A glass cladding with a lower index of refraction than the core surrounds the glass core, to preserve all the light in the core. Finally, to protect the cladding a thin plastic jacket is used and the fibers are generally organized into groups of bundles, which is protected by using an outer sheath. A sheath with three

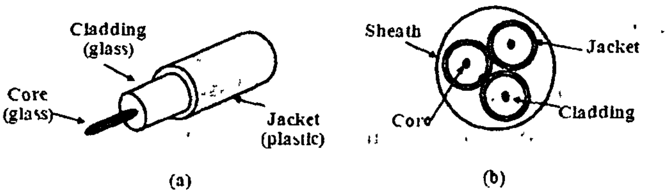


Figure 2.3: (a) Side view of a single fiber. (b) End view of a sheath with three fibers

fibers is shown in Figure 2.3 (b).

2.1.2.2 Unguided Media

The unguided signals can travel from the transmitting end to the receiving end in several ways. As reported in Table 2.1, such media can propagate in all directions and they are relatively less costly and their installation is also less time consuming. In this section we shall discuss mainly four types of communication medium, namely (i) *Radio Transmissions*, (ii) *Microwave Transmission*, (iii)*Lightwave Transmission* and (iv) *Infrared* and *Millimeter Waves*.

Radio Transmission

This unguided communication medium is commonly known for both indoor as well as outdoor applications because of their: (i) easy generation, (ii) capacity to travel long distances, and (iii) capacity to penetrate buildings easily. Unlike guided media, such waves are frequency dependent and can propagate in all directions (omnidirectional) from the source, which simplifies the task of alignment between the transmitter and the receiver. A radio wave with low frequency can pass through obstacles easily, however, with the increase in distance the power falls off sharply. On the other hand, a high frequency radio wave can easily travel in straight lines and bounce off obstacles. Generally, frequencies in the range (30 MHz-1 GHz) are referred as the *radio range*, which are typically suitable for any omnidirectional applications.

Microwave Transmission

This type of unguided medium is narrowly focused; high frequency (1 GHz-40GHz) waves, travel in nearly straight lines. At these frequencies, it is quite possible for point-to-point microwave transmission since the beams are highly directional. However, as these waves are highly directional, if the transmitting towers are too far apart, then there's every possibility that the earth will get in the way. As a result, it demands for the repeaters periodically With the increase in the height of the towers, they can be placed apart accordingly However, the microwaves cannot pass through buildings well, as can be found in case of low frequency radio waves.

Lightwave Transmission

These are inherently unidirectional communication medium. It is used to connect networks in different buildings through roof-mounted lasers. Due to its unidirectional nature, each building needs to have its own laser as well as its own detection mechanism (e.g., photodetector). This cost-effective medium is well known for its high bandwidth.

Infrared and Millimeter Waves

This medium is widely used for short-range *point-to-point* and *multipoint* applications. Presently, almost all the electronic devices such as televisions, VCRs, VCDs, and stereos are operated with infrared communication. The frequency range from 3×10^{11} to 2×10^{14} Hz is referred as Infrared. They are relatively directional, easy and convenient to use, cheap, and easy to build. However, a major limitation is that they cannot pass through solid objects.

2.1.3 Network Software

In the initial development of computer networks, hardware was the main concern and the software was added later. However, this strategy does not work longer. To provide more and more functionality, sophistication and structureness have been introduced in software. In the subsequent sections we discuss the structureness in the network software in some detail.

2.1.3.1 Layered Architecture

To reduce the design complexity, most networks are organized in a layer-wise or level-wise fashion, where the top layer is designed based upon the layer below it. Here, each layer is designed with an objective to offer a set of pre-defined services to the layer above it, hiding the details regarding the implementations of those offered services. During the conversations between two layers, say layer-n and layer-$(n - 1)$, the rules and conventions used,

Table 2.2: Six different types of services

S.N.	Services	Type	Example
1	Reliable message stream	Connection-oriented	Sequence of pages
2	Reliable byte stream	Connection-oriented	Remote login
3	Unreliable connection	Connection-oriented	Digitized voice
4	Unreliable datagram	Connectionless	Electronic junk mail
5	Acknowledged datagram	Connectionless	Registered mail
6	Request-reply	Connectionless	Database query

are collectively referred as the layer-n protocol. A protocol represents an agreement on the communications to be proceed between two parties.

2.1.3.2 Connection-Oriented and Connectionless Services

Layers are designed to offer two types of network services : *connection-oriented* and *connectionless.* In *connection-oriented* service, the user (i) establishes a connection, (ii) uses the connection based on a mutually negotiated (the sender/receiver/and the subnet) parameters, e.g., maximum size of message, required quality of service, and other relevant issues, and (iii) releases the connection In case of *connectionless* services, there is no logical connections through the network on which data flow can be regulated, as a result the data transfer becomes unreliable. Analogous to telegram service, an unreliable or unacknowledged connectionless service is referred as datagram service. An example of datagram service is the request-reply service. Table 2.2 lists various types of services[50] and their examples.

2.1.3.3 Service Primitives

Primitives are the operations. A service is specified by a set of primitives. A user process can access these primitives to access the service. Connection-oriented service-primitives are different from the primitives for connectionless service. Table 2.3 shows a list of service primitives used in a client-server environment to implement reliable byte stream.

Table 2.3: Services Primitives

S.N.	Name	Purpose & Action
1	LISTEN	to accept an incoming connection; server process is blocked till connection request arises;
2	CONNECT	to establish a connection ; client process is suspended till there's a response;
3	RECEIVE	to receive the request of the client; block wait for a message;
4	SEND	to send a message; once packet reached, unblocks the server process;
5	DISCONNECT	to terminate a connection; once packet received, server acknowledges;

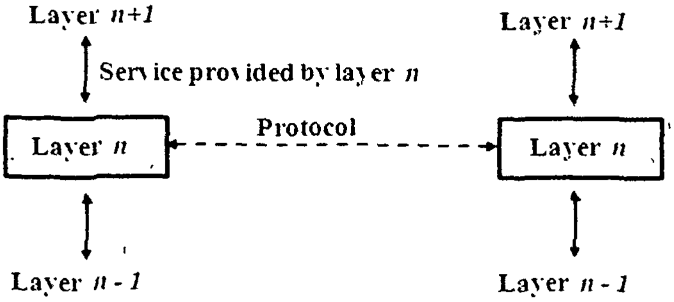


Figure 2.4: The relationship between service and protocol

2.1.3.4 Services and Protocols Relationship

Services refer to a set of operations provided by a layer to its immediate upper layer. It also relates to an interface between two consecutive or adjacent layers, where the bottom layer serves as the service provider and the upper layer acts as the service user. Services related to the interfaces between layers are illustrated in Figure 2.4. A protocol is a set of rules used to govern the meaning and format of the packets, or messages exchanged between the peer entities within a layer. Entities at the same layer as shown in Figure 2.4 referred as peer entities. Protocols are used by the entities to implement their services. Ensuring their predecided services, they are free to change the protocol.

2.1.4 Reference Models

After the introduction of some of the basics of networking, we shall now discuss on an important topic i.e., network architectures or reference models and its two well known types : (i) the OSI reference model and (ii) the TCP/IP reference model. Let's begin our discussion with the OSI model

2.1.4.1 The ISO OSI Reference Model

This pioneering Open Systems Interconnection (OSI) model was introduced by the International Organization for Standardization (ISO) as a reference model for computer protocol architecture and as a framework to support developing protocol standards. This model comprises of seven layers as illustrated in Figure 2.5 Here, dotted lines are used to represent the virtual communication and with the solid lines the physical communication is shown. A brief discussion on the purpose of each layer is reported next.

Application layer: This layer provides the users: (i) access to the OSI environment and (ii) distributed information services.

Presentation layer: This layer intends to provide independence to the various application processes from differences in data representation (syntax).

Session layer: It provides the control structure for communication between applications. Establishment, management and termination of connections between cooperating applications are the major responsibilities of this layer.

Transport layer: This layer supports a reliable and transparent transfer of data between two end points. It also supports end-to-end error recovery and flow control.

Network layer: This layer supports its upper layers with independence from the data transmission and switching technologies used to connect systems. It is also responsible for the establishment, maintenance and termination of connections.

Data link layer: The responsibility of transferring the information across the physical link reliably is assigned to this layer. It transfers blocks (frames) with the necessary synchronization, error control, and flow control.

Physical layer: This layer is responsible for transmitting a stream of unstructured bits over physical medium. It deals with the mechanical, electrical, functional, and procedural characteristics to access the physical medium.

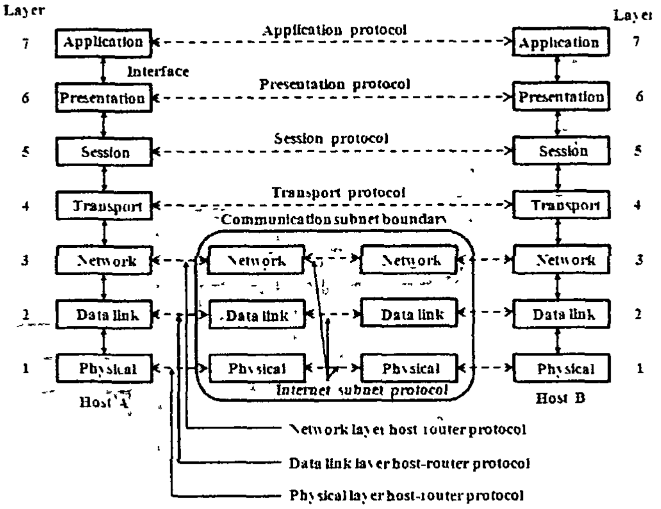


Figure 2.5: The OSI reference model

The interfacing between each pair of adjacent layers is defined in terms of a set of pre-defined primitive operations and services made available by a layer to its immediate upper layer.

2.1.4.2 TCP/IP Reference Model

Over the decade, researchers have made several significant efforts to introduce variants (extended or compact versions) of the OSI reference model, however, the overall seven-layer model has not flourished. In an attempt to introduce a cost-effective version of the OSI model, the TCP/IP model[51] was introduced with fewer layers, which has been ultimately accepted widely by the network community. The functions of TCP/IP network architecture are divided into five layers. Figure 2.6 shows all these layers of the TCP/IP architecture. Networking hardwares and softwares are organized into different layers based upon the functions of the routers and bridges. However, the functions of all the layers are not supported by the routers and bridges, for example, functions of layer 1 & 2 are supported by the bridges, and layers 1 through 3 are implemented by the routers, as shown in Figure 2.6.

The *application layer* is responsible for supporting network applications. It is well known to all that with the evolution of computer and networking

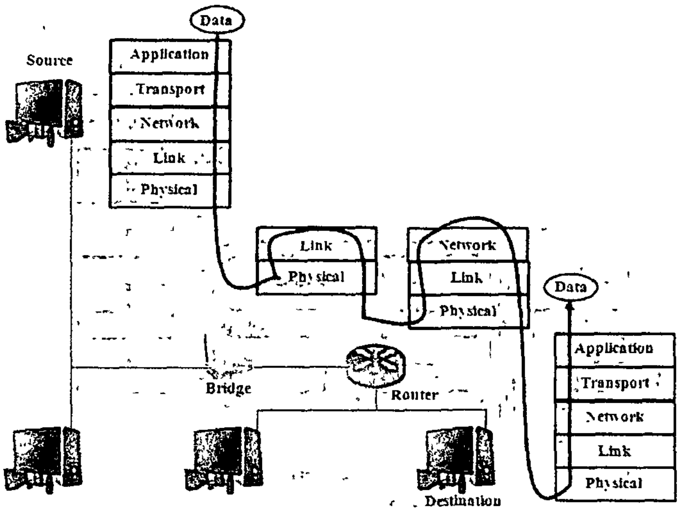


Figure 2.6: TCP/IP Architecture

technologies, the types of applications to be supported by this layer are also changing. It requires a separate module dedicated for each application type, such as file transfer. The *application layer* includes many protocols for various purposes, such as (i) to support the Web : Hyper Text Transport Protocol (HTTP), (ii) to support electronic mailing: Simple Mail Transfer Protocol (SMTP), and (iii) to support file transfer FTP (File Transfer Protocol) .

The *transport layer* is responsible for transporting *application layer* messages between the client and server. It plays a very crucial role in the network with the help of two transport protocols, TCP and UDP. Both these protocols support transportation of *application layer* messages either through connection-oriented or through connectionless service. TCP supports a guaranteed connection-oriented service to deliver *application layer* messages to the destination. TCP also provides a congestion control mechanism by segmenting a long message into shorter segments, so that during the network congestion a source can throttle its transmission rate. A connectionless, unreliable and non-guaranteed service of message delivery is provided by the UDP protocol to its applications by simply sending packets.

The *network layer* provides the service of routing datagrams among the hosts. It uses specialized hosts, called gateways or routers to forward packets in a network. The router selects paths to forward the packets towards their destination. The *network layer* support IP protocol (Internet Protocol) that defines the fields in the IP datagram that determine the working of end systems and routers. To determine the routes for forwarding the packets between the sources and destinations, this layer also includes routing protocols. The TCP or UDP protocol in a source host forwards a transport layer segment and a destination address to the next layer protocol, i.e., IP. It is now responsibility of IP layer to deliver the segment to the destination host. On arrival of the packet at the destination host, IP passes the segment to the transport layer within the host. The *network layer* performs the routing of a packet through a sequence of packet switches between a pair of source and destination.

The *network layer* is dependent on the services of the *link layer* while forwarding a packet in the route from one node (host or packet switch) to the next node. In particular, at each node, the *network layer* passes the datagram down to the *link layer*, which delivers the datagram to the next node along the route. At this next node, the *link layer* passes the datagram up to the network layer depending on the specific *link layer* protocol that is employed over the link. For example, a datagram may be handled by various protocols at different links, e.g., Ethernet on one link and then PPP (Point-to-Point Protocol) on the next link. The *network layer* will receive a different service from each of these different protocols (*link layer*).

The *physical layer* provides service to move the individual bits of a frame from one node to the other. So, the protocols included in this layer are link dependent and also depends on the physical transmission medium of the link (e.g., co-axial cable, twisted pair, etc).

2.1.5 Protocols

A protocol is a set of rules used to govern the meaning and format of the packets, or messages exchanged between two or more peer entities, as well as the actions taken on the transmission and/or receipt of a message or other event. Protocols are extensively used by computer networks.

Transport Control Protocol (TCP)

This transport layer protocol supports a reliable connection for transferring data between applications. A connection is basically a logical association established temporarily between two entities in different systems. The logical

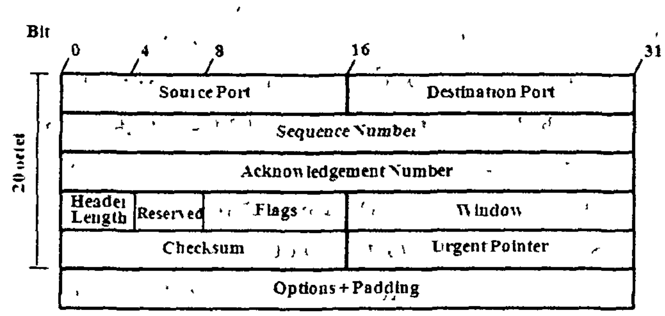


Figure 2.7: TCP header

connection or association is here referred by a pair of port values. To regulate the flow of segments properly and to recover from damaged or lost segments, entity keeps track of TCP segments moving (both way) to the other entity during the connection. A TCP header format of minimum 160-bit is shown in Figure 2.7. The source port and the destination port shown at the top of the header are used to gather information about the applications running at the source and destination systems currently in connection. The other fields in the header such as *Sequence Number, Acknowledgment Number*, and *Window* fields are used for flow and error control. To detect occurrence of errors in the TCP segment, the checksum field (a 16-bit sequence) is used.

User Datagram Protocol (UDP)

This connectionless protocol provide services of message delivery, sequence preservation, or duplication protection. However, it does not guarantee of these services UDP supports a simplified protocol mechanism to enable a procedure to send messages to other procedures. This protocol you can find in use among the transaction-oriented applications, such as SNMP (Simple Network Management Protocol), the standard network management protocol for TCP/IP networks. Due to its connectionless characteristics, UDP has a minimum role to play It basically adds a capability of port addressing to IP, which can be understood by observing the simple UDP header, as shown in Figure 2.8. Like TCP, it also includes a checksum field (optional) to'verify the occurrence of error(s) in the data

Internet Protocol (IP)

This IP layer protocol is responsible for routing across multiple networks A minimum of 160-bit(20 octets) IP header format (of IPv4) is shown in

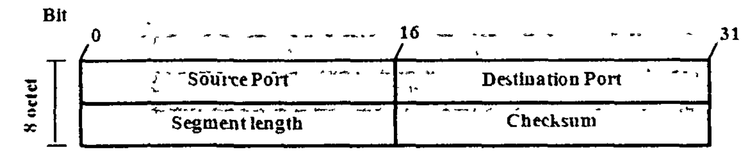
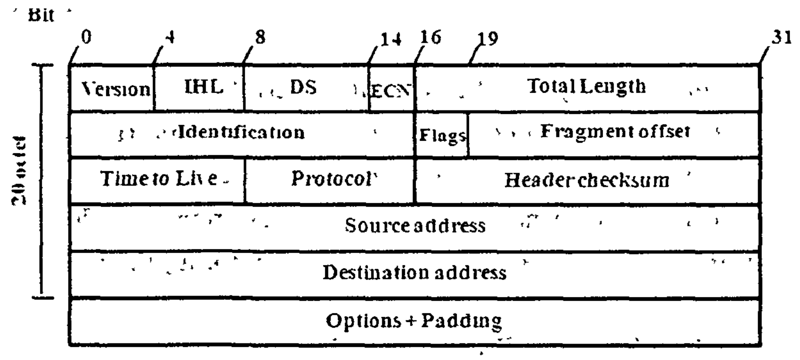


Figure 2.8: UDP header



IHL = Internet Header Length DS = Differentiated services field
ECN = Explicit congestion notification field

Figure 2.9: IPv4 header

Figure 2.9. An IP packet or datagram (i.e. IP-level PDU) is constituted by combining the header along with the segment, from the transport layer. The source and destination addresses in the IP-header are 32-bit; also it includes a checksum field to help identifying occurrence of errors in the header to avoid wrong delivery. The Protocol field is used to indicate the higher-layer protocol using the IP, similarly, the ID, Flags, and Fragment Offset fields are used in the fragmentation and reassembly process.

Simple Mail Transfer Protocol (SMTP)

Today, SMTP is an integral component of the modern electronic mailing system. It offers as a basic electronic mailing facility, which supports transferring of messages among separate hosts. Some of the salient features of SMTP are : (i) mailing lists, (ii) sending back receipts, and (iii) message forwarding. However, the message creation is not under the control of SMTP protocol; user can take the support of local editing or electronic mail facility. Once message creation task is over, SMTP forwards the message with the help of TCP to another SMTP module on an host. The incoming message will be

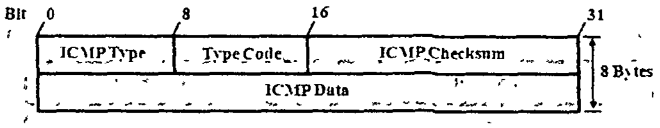


Figure 2.10: ICMP header

received by the target SMTP module and will store it in a user's mailbox by using a local mailing package .

Simple Network Management Protocol (SNMP)

This application-layer protocol is designed with an objective to provide facility to forward and receive management information among the network devices. It facilitates to manage the entire network from a single point by assigning an IP address to each switch, and by monitoring the interfaces on that switch.

Internet Control Message Protocol (ICMP)

The responsibility of ICMP (RFC 792) is to provide a means for forwarding messages from router(s)/host(s) to other host(s). It supports a feedback mechanism to inform about the problems in the communication environment. We can have several applications of this protocol, like:(i) In a situation, when a datagram is unable to reach its destination, (ii) Lack of buffering capacity in a router to transfer a datagram, and (iii) When the router is in a position to direct the station to forward traffic on a short-distance route Generally, a message of this protocol is sent either by a router along the datagram's path or by the intended destination host, in response to a datagram. Figure 2.10 shows the format of an ICMP header. It is atleast of 8 Bytes header. Various fields of an ICMP header are : (i) 8-bit *type* to represent the type or category of an ICMP message, (ii) 8 bits *type code* to specify parameters of the message that can be represented in one or a few bits (iii) 16 bits *checksum* of the entire ICMP message, like that used for IP, and (iv) 32 bits *ICMP Data* to specify lengthy parameters.

File Transfer Protocol (FTP)

It is a very commonly used protocol to transmit files from one system to another based on user command. FTP supports transmitting both text and binary files Also, it facilitates user for controlled access. On invoking the FTP for file transfer, a TCP connection is established with the target system to exchange the control messages. This TCP connection is used to transmit user ID and password and then the user is asked to specify the file(s) to be transmitted and the desired file actions. Based on approval of the file transfer, this protocol initiates to set up a second TCP connection for transferring the data. Now, using the data connection the desired file is transferred to the target system, without any overhead of any headers or control information. Once successfully the transfer is over, a completion signal is indicated by the control connection.

Telnet

Telnet is short for 'Terminal Emulation', which provides users a remote login capability. It enables a terminal user or a client to login to any remote computer or server. Once logged in, the user can function as a directly connected user of that computer. The design of this protocol facilitates the user to work as if in a local terminal with simple scroll-mode facility. Telnet is implemented in two distinct modules with distinct characteristics: (i) *User Telnet*, which enables to interact with the I/O module of the terminal to communicate with a local terminal by mapping the characteristics of a real terminal to the network standard and vice versa. (ii) *Server Telnet*, which acts as a surrogate terminal handler to interact with an application, so that the behaviour of the remote terminal looks like local to the application. Transmission of traffic between User and Server Telnets occurs using a TCP connection.

2.1.6 Types of Networks

There are many many different kinds of networks, large and small. They have different goals, scales, and technologies.

Local Area Networks (LAN)

LANs are privately-owned, limited area network that interconnects a number of computers within a single building, such as home, institution or organization, or a campus of up to say, 1 kilometer in size. These LANs are commonly used to connect the machines in offices, institutions or any organizations to share and exchange e-resources (e.g., digital documents, printers, scanners, etc). The three distinguishing features of LAN are: (1) size of coverage area, (2) communication technology used, and (3) use of network topology. LANs may be developed by using different (i) communication media types,

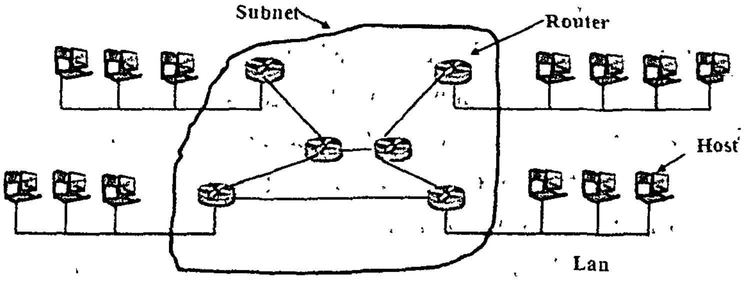


Figure 2 11: Relation between hosts on LANs and the subnet

such as guided or unguided media, and (ii) network topologies, such as star, ring, bus, etc. With the increasing development of computer and communication technologies, the growth of speed of a network is also increasing at a phenomenal rate. The speed of traditional LANs are between 10 Mbps to 100 Mbps, with minimum delay (microseconds or nanoseconds). However, in the recent LANs, the achievable speed is upto 10 Gbps, where 1 Gbps stands for 1,000,000,000 bits per sec.

Wide Area Networks (WAN)

WANs are created by interconnecting a large number of machines (or hosts) spanning across a broad geographical area, often a country or continent or a region by using communication subnet (private or public). The communication subnet may be organized based on two basic principles to transport messages from one machine (or host) to other: *store-and-forward* or *packet-switched*. Here, each subnet basically consists of two components: *switching components* and *transmission media*. Switching components are specialized and intelligent devices, referred as routers. As discussed in the subsection 2.1.2, media can be of various types such as coaxial cables, twisted copper wires, optical fibers, or microwave transmissions to forwards bits among the connecting machines. On arrival of packets on an incoming line, the switching component selects an outgoing line to forward the packet. Figure 2.11 shows a model of WAN. In a WAN, a host may be either connected to a router directly, or may be frequently connected to a LAN on which a router is present

Metropolitan Area Network(MAN)

A MAN may be created by connecting several local area networks or Campus LANs spanning a township, city, or metropolitan area. Due to its wide area coverage requirement for sharing and exchanging e-resources, a MAN includes several routers, switches and hubs. To provide faster wireless Internet access, MAN is created using IEEE 802.16 standard.

Wireless Networks

Like wired networks as discussed above- the wireless networks are also can be of various types depending on its coverage area. Their primary objective is to provide remote information transmission facilities by using unguided media as carrier, such radio waves. The three basic types of wireless networks are:
(a) *System interconnection*: It is an interconnection of the various components of a computer viz., monitor, keyboard, mouse, and printer using short-range radio in personnel area network.
(b) *Wireless LANs*: Here, a system communicates with the rest of the systems within a building or an office area, by using a radio modem and antenna. Today, such wireless LANs are common in almost all the modern low and medium-scale corporate offices. Generally, a wireless LAN is convenient to create in those places where installing Ethernet is more troublesome.
(c) *Wireless WANs*: This is an extended (scope and area of coverage-wise) version of wireless LAN and equivalent with its wired counterpart i e. WAN. An example of wireless WAN is cellular networks.

Internetworks

An internetwork is created by connecting a network with several other networks (may be incompatible) using a specialized device.or machine,.called gateway. Two important requirements in the formation of an internetwork are: (i) To connect incompatible or hybrid networks, and (ii) To provide adequate translation, both in terms of hardware and software. A common example of an internetwork is a WAN connected with a collection of LANs.

The Internet

The Internet is a worldwide network of computers, that interconnects billions of users or computing devices spanning across the globe. The Internet is based on TCP/IP protocol (however, all applications may not use TCP) to connect its end systems (or computing devices). The end systems are interconnected via both guided or unguided transmission media. The transmission rate of a communication link is often referred as the *bandwidth* of the link, which is usually quantified in terms *bits per second* or *bps*. All the computing devices or end systems are interconnected (indirectly) through intermediate switching devices called *routers*. On arrival of a chunk of information on one of its incoming links, the *router* forwards it on any of its outgoing links. The path chosen for forwarding the packet from the sending end to the receiving end system through a sequence of links and routers is known as the *route* or *path* across the network. For cost-effective utilization of the resources while forwarding a packet, the Internet uses a technique known as *packet switching* that allows to share the same path (or parts of a path) among multiple communicating end systems concurrently. To provide access to the end systems of the Internet, a service provider, called Internet Service Providers (ISPs) is maintained, which is basically an organization of routers and communication links from a range of technologies to facilitate end-users of the Internet access. The different ISPs provide a variety of different types of network access to the end systems, including high-speed LAN and wireless access.

2.1.7 Scale of Networks

Apart from those characteristics discussed above to classify a network, another way to classify a network is its scale. Based on the inter-processor distances and their span or coverage, networks can be classified into six distinct classes, as shown in Table 2.4. Small scale personal area networks, meant for single user are shown on the top rows. Example of such a network is a wireless network connecting a computer with its keyboard, mouse and printer. Below personal area networks, various types of long-range networks are shown. These networks are classified into three distinct sub-classes based on their range or coverage: (i)Local area network or LAN, (ii) Metropolitan area network or MAN and (iii) Wide area networks or WAN. A LAN may cover within a 1 km range, whereas a WAN may spread over a country or a continent i.e., may be upto 1000 km range. Finally, the inter-connection of several networks involving billions of users or computing devices spanning across the globe, is called an internetwork. A well-known example of an internetwork is the worldwide Internet. Such classification of networks based on distance is important, specially from the applicability of the different techniques.

Table 2.4: Classification of interconnected processors by scale.

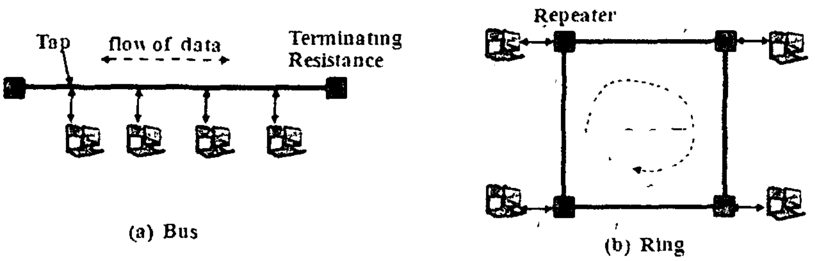
Interprocess Distances	Processors Located in	Example Network
0.1m	Same circuit board	Data flow machines
1m	Same system	Personal Area Network
10m - 1km	Same room, building or campus	Local Area Network
10km	Same city	Metropolitan Area Network
100-1000km	Same country or continent	Wide Area Network
10000km	Same planet	The Internet

2.1.8 Network Topologies

Network topologies are the various arrangements or ways of interconnecting the end systems or computing devices attached to the network. Topologies can be of two types: *physical* and *logical*. Physical topology basically refers to the geometric shape of the layout of the physical media used in the network.. Whereas, logical topology represents the way data flows from one end system to another in the network. The logical topology of a network may not be same with its physical topology. Logical topologies can be dynamically re-configured using special device called router. Eight types of topologies are available in the network topology literature: (a) Point-to-point, (b) bus, (c) ring, (d) star, (e) tree, (f) mesh, (g) and (h) hybrid. In this section, we shall discuss four popular types of network topologies, i.e. bus, ring, tree and star.

Bus

In a LAN, for this topology, all stations of the network are attached directly to a linear transmission medium or the bus, through an appropriate hardware interfacing known as a tap. Here, the data flows from the source machine to all other machines in both the directions until the target recipient is found. Bus can be of two types: *linear* and *distributed*. In case of *linear bus*, the transmission medium has exactly two end points, and it enables all the nodes in the network to receive the transmitted data simultaneously. Whereas, in case of distributed bus, there can be more than two end points, created by adding more branches to the main transmission medium. In case of simultaneous transmission of data by multiple machines, an additional



(a) Bus

(b) Ring

Figure 2.12: Two broadcast topology of networks:(a)Bus,(b)Ring.

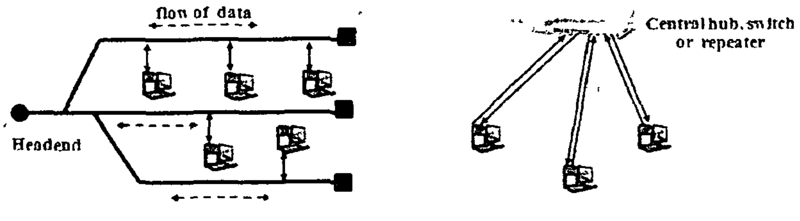
arbitration mechanism is necessitated in order to resolve conflicts. The operation of this mechanism may be *centralized* or *distributed*. For example, Ethernet (IEEE 802.3) is a decentralized bus-based broadcast network operating within a range of (10 Mbps-10 Gbps). Example of a bus topology of network is shown in Figure 2.12 (a).

Ring

In this topology, a set of devices (act as repeaters, capable of receiving and transmitting data bit by bit) are connected in a circular fashion by point-to-point links in order to maintain the signal strength. Here, the transmission of data (frames) is unidirectional (clockwise or anti-clockwise). As a frame moves in the circular path, the destination repeater recognizes its address and copies the frame into a local buffer. The transmission of the frame continues till it returns to the source repeater, where it is eliminated. Since the ring is shared by multiple stations, medium access control has a responsibility to determine time for each station to insert frames. Two common examples of this topology are: (a) IEEE 802.5 token ring, which operates at (4-16) Mbps, and (b) FDDI (Fiber Distributed Data Interface) ring network. An example ring topology of network is shown in Figure 2.12 (b).

Tree

Here, the arrangement of inter-connection is made in a tree-like fashion, i.e., stations (which have *unique addresses*) are interconnected to a transmission medium with branches without a loop. The beginning point of the tree is referred as the *headend*, from where multiple branching cables can be initiated, and each these branches can have additional branches which may



(a) Tree

(b) Star

Figure 2.13: Two broadcast topology of networks:(a)Tree,(b)Star.

lead to a complex layout. Again, data is transmitted in small blocks, referred as *frames*, from any station towards all other stations throughout the transmission medium. Each of these frames has two basic components: (i) frame header, containing the control information, and (ii) a portion of the data intended to be transmitted by a station. Header of a frame also includes the destination address for the frame. An example of tree topology network is shown in Figure 2.13 (a).

Star

Here, the stations are arranged in a star fashion with a common central node, connecting all the stations in the network. A star topology network is shown in Figure 2.13 (b). The stations are attached to a central node using two point-to-point links for transmission as well as for reception. This topology allows two alternatives for the central node to operate: (i) Operate in a broadcast fashion, where a frame transmitted from one station to the node (often referred to as a *hub*), is retransmitted on all of the outgoing links. (ii) Operates as a frame switching device, where the central node buffers an incoming frame and then it retransmits on an outgoing link to the destination station.

2.1.9 Hardware Components

A collection of softwares and hardware components are necessary to interface with the communication medium and to regulate the organized access to the medium in computer networking.

2.1.9.1 Network Communication Devices

A. Repeater and Hubs

The purpose of a repeater is to receive, amplify and retransmit signals bidirectionally. Here, from software point of view, connecting a series of cable segments by this physical layer device makes no significant difference from a single cable, however, some amount of delay may occur. By allowing multiple cables to connect with a repeater, it enables to create larger networks.

Unlike repeater, generally a hub does not amplify the incoming signals. It includes several incoming lines joined electrically. It allows the frames arriving on an incoming line, to transmit out on all the others. In case of collision (i.e., arrival of two frames at the same time), the entire hub forms a single collision domain. An important requirement of this device is that-speed of incoming signals into a hub must be same. Three basic types of hubs are: (a) *Active hub*: It is capable of performing like a repeater, can also regenerate or amplify the signals. This type of hub requires electrical power to run and can cover upto 2000 feet. (b) *Passive hub*: This kind of hub acts as a central device, forwards signals to other devices connected to it. It allows to connect multiple stations in a star fashion. A passive hub can cover upto 300 feet. (c) *Intelligent hub*: This advanced type of hubs have also management capability. Apart from signal regeneration, it is also capable of performing network management and intelligent path selection An intelligent hub has ability to offer flexible transmission rates to various devices.

B. Bridges

This device works at data link layer to provide connectivity between LANs. It supports routing by examining the data layer link addresses. Bridges can transport IPv4 or any other kinds of packets, as they do not examine payload field. It minimizes the processing overhead by allowing to use only same protocol. Without any additional burden on the communications software, this device provides an extension to the LAN. Bridges are of two types: (a) *Transparent bridge*: The presence and operation of this type of bridge are made transparent to these network hosts. It analyzes the pattern of the incoming source addresses from associated networks and gather knowledge about the topology of it. By using its internal table, the bridge forwards the traffic. This kind of bridges are standardized into the IEEE 802.1 and are popular in Ethernet/IEEE 802.3 networks. *Routing Bridge (RBridge)*: The operation of this device is based on the TRansparent Interconnection of Lots of Links (TRILL) protocol. This protocol (TRILL) supports (i) optimal pair-wise data frame forwarding service without configuration, (ii) safe forwarding even during the periods of temporary loops, and (iii) multipathing of both unicast and multicast traffic. To provide these services, TRILL uses a link-state protocol, referred as IS-IS (Intermediate System to Intermediate System) protocol. The IS-IS protocol supports broadcast connectivity to all the RBridges, to know each other and their connectivity. Based on these information, a RBridges could compute (a) pair-wise optimal paths for unicast, and (b) distribution trees for delivery of frames either to destinations or to multicast/broadcast groups.

C. Switches

This network device works at data link layer to provide a dedicated connection between ports. It minimizes the transmission overhead by determining which ports are communicating directly and connects them together. They interpret and operate upon the MAC address in the receiving packets. On arrival of a packet at a port, this device makes a note of the source MAC address, associates it with that port and stores this information in an internal MAC table. Based on the available destination MAC address in its MAC table, the device finally forwards the packet on the respective port. If the MAC table doesn't include the destination MAC address, packet is transmitted to all the connected interfaces. However, in case of a match between the destination port and the incoming port, the packet is simply filtered. Figure 2.14 shows working of a switch. Two basic types of switches are: (a) *Store-and-forward switch*: It follows a three-step execution policy: accept, store and forward. Here, a frame is accepted on an input line, buffers it, and then transmits it to the appropriate output line. (b) *Cut-through switch*: This type of switches take the advantage of the occurrence of the destination address at the beginning of the MAC frame. It attempts to recognize the destination address by repeating the incoming frame onto the appropriate output line.

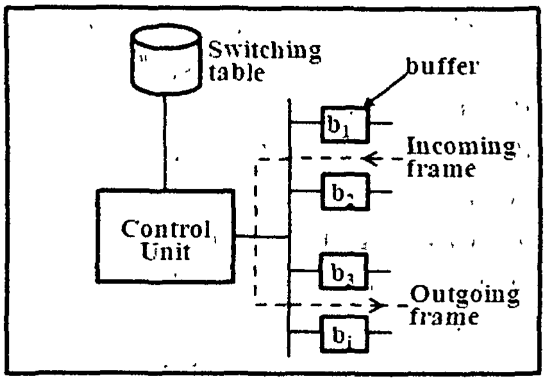


Figure 2.14: Working of a switch

2-layer switch

The operation of this category of switch is almost like a full-duplex hub. It also provides scopes to include additional logic to use it as a multiport bridge. By increasing the capacity of a layer-2 switch, several additional devices can be hooked up with it, and it can keep up with all attached devices. It performs the address recognition and frame forwarding functions at the hardware level, and can handle multiple frames simultaneously.

3-layer switch

To address the two important limitations of Layer 2 switches, viz, (i) broadcast overload and (ii) lack of multiple links support, a 3-layer switch came into existence. It breaks a large LAN logically into several sub-networks connected by a special network device, called router (discussed in the subsequent section). Based on the operation, a Layer 3 switch can be of two categories: *packet-by-packet* and *flow based*. The first category of layer-3 switch operates similar like a traditional router. Whereas, the other category switch attempts to improve its performance by grouping IP packet flows with identical source and destination. Generally, layer-3 switches are interconnected at 1 Gbps and connected to layer 2 switches at 100 Mbps-1 Gbps.

D. Routers

The main objective of this device is to connect two networks by relaying data from a source to a destination end system based on a routing table that includes lists for each possible destination network the next router to which the internet datagram should be sent. A routing table can be static as well as

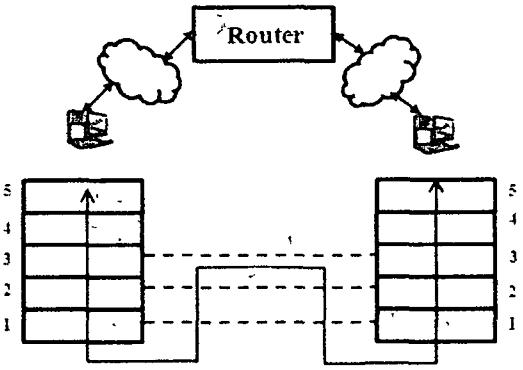


Figure 2.15: Router in the TCP/IP model

dynamic. In static routing table, alternate routes are maintained to handle the unavailability of a particular router. However, a dynamic routing table provides more flexibility in response to both error and congestion conditions. Such a router can support various types of network media, such as Ethernet, ATM, DSL, or dial-up. Apart from those dedicated routers, sch as Cisco routers, it can also be configured using a standard PC with several network interface cards and appropriate software. Generally, routers are placed at the edge of multiple networks. Usually, they have a connection to each network, and also they can take on other responsibilities as well as routing. Many router are also equipped with firewall capabilities to filter or redirect unauthorized packets. Also, most routers are capable to provide Network Address Translation (NAT) services. Figure 2.15 shows a router in the TCP/IP model.

E. Gateway

A gateway serves as a server on behalf of other servers which may not have a direct communications with a client. Two major services provided by this device are: (i) as server-side portals through network firewalls, and (ii) to provide access to resources stored on non-HTTP systems as a protocol translators. It acts as an original server to receive requests from clients for the requested resources, without the knowledge of the client about it.

2.1.9.2 Network Interface Card (NIC)

A. Ethernet Card

Ethernet has already been established as a well-accepted standard (officially called IEEE 802.3) to connect together the stations on a LAN. It can also be used as an interface to connect a desktop PC to the Internet via a router, or an ADSL modem (Asymmetric Digital Subscriber Line). A most commonly known Ethernet standard is 100baseT, which represents a transmission media type e.g., twisted pair cables with modular RJ-45 connectors on the end, allowing to transmit data at the rate of 100 Megabits/second. This standard supports the star network topology, with switches or hubs at the central node, and the end devices (or additional switches) at the edges. Here, each device connected to an Ethernet network is assigned a unique MAC address given by the manufacturer of the NIC card, which functions like an IP address.

B. LocalTalk

It is a special kind of physical layer implementation of the AppleTalk networking system. It generally uses the bus topology with the devices connected in a daisy chain fashion. LocalTalk uses the bus system of shielded twisted pair cabling, plugged into self-terminating transceivers, running at a rate of 230.4 Kbits per second. However, with the widespread popularity of Ethernet-based networking has out-numbered the LocalTalk.

C. Connector

To provide a physical link between two network components this is a widely used device. To connect electrical circuits, an electrical connector is used, which is basically an electro-mechanical device. Based on the requirements, this connection either may be temporary or permanent. For portable equipments, the connection may be temporary, which usually demand for an arrangement for assembly and removal. On the other hand, a permanent joint is needed for connecting two wires or devices. Depending on the requirements, connections of various types can be found in use. Some of the example features include: shape, size, gender, connection mechanism and function. Some of the commonly found connectors are : Power connectors, USB connectors, D-subminiature connectors, 8P8C connector, BNC connectors, etc.

D. Token Ring Cards

Two well-known Token-Ring examples are : IBM's Token Ring network and IEEE 802.5 networks. In a Token Ring network, the token is allowed to move between the computers inside a logical ring and the physical layout of the cable ring passes through the hub As a part of the ring, the users are connected to the ring via the hub. It supports a ring topology, however, in some special implementation, it may use a star-wired ring topology with all the computers on the network connected to a central hub (for example, the IBM's Token ring implementation). The transmitted frame moves around the ring until the intended destination station is reached. However, if the transmitted frame couldn't find the destination station, and finally reaches the sending station again, it is eliminated. It is the responsibility of the sending station to verify the returning frame whether it was seen and subsequently copied by the destination or not. This architecture is considered to be stable and demanding due to its ability to handle high-bandwidth video conference and multimedia applications. A typical transmission speeds of this standard are of 4 Mbps or 16 Mbps.

2.1.9.3 Transceivers

The main purpose of a transceiver is to clamp around the cable securely so that its tap makes contact with the inner core properly. A transceiver is equipped with the electronics to handle carrier detection and collision detection. On detecting collisions, a transceiver typically puts an invalid signal on the cable for other transceivers to realize that a collision has occurred. The cable of the transceiver is terminated on a computer's interface board, where the interface board includes a controller chip that transmits frames to, and receives frames from, the transceiver. Two major responsibilities of the transceiver are: (i) to assemble the data into the proper frame format, and (ii) to compute checksums on outgoing frames to verify them on incoming frames.

2.1.9.4 Media Converter

This network device supports connectivity between two dissimilar media types such as twisted coper wires with fiber optic cabling. In several real-life networking applications, it becomes essential to interconnect an existing copper-

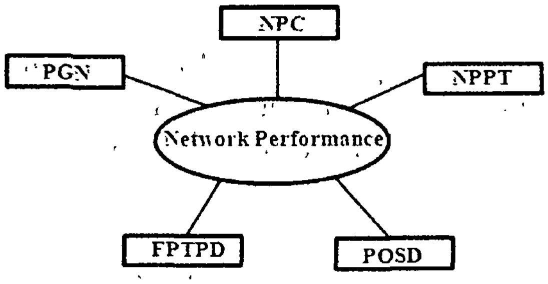


Figure 2.16: Various aspects of Network Performance

based, structured cabling system with fiber optic cabling-based systems. This device supports electrical-to-optical connectivity (with conversion) typically at the OSI Layer-1, i.e. physical layer, to bridge the difference between copper and fiber optic cabling. Media converters are found in use mostly in the enterprize networking to provide the clients data transport services.

2.1.10 Network Performance

In a network with hundreds or thousands of nodes or hosts, complex interactions with unforseen consequences are common. Such complexity often leads to performance degradations. So, appropriate tuning of the crucial performance parameters is a must. Next, we shall discuss five important aspects of network performance, as shown in Figure 2.16.

2.1.10.1 Network Performance Constraints (NPC)

Many times, the performance of network suffers due to temporary resource overloads. The sudden increase in the traffic arrival at a router beyond its capacity, congestion may build up and performance may suffer. Another important constraint of network performance degradation is resource imbalance. For example, if a gigabit communication line is attached to a low-end PC, the poor CPU will not be able to process the incoming packets fast enough and some of them will be lost. As a result, these packets will eventually be retransmitted, causing delay, wasting bandwidth, and finally will result with *degraded performance. Another performance constraints is due to lack of* a proper parameter tuning. For example, if a TPDU (Transport Protocol Data Unit) contains a bad parameter (e.g., the port for which it is destined), often the receiver will suspect and send back an error notification. This is single instance, now imagine what could happen, if a bad TPDU is broadcasted to 10,000 machines: each one might send back an error message. Such a broadcast storm could cripple the network.

2.1.10.2 Network Performance Parameter Tuning (NPPT)

A significant step towards improvement of a network performance is to understand the happenings in the network by appropriate performance measurements. Once the relevant parameters are identified, one should try to understand the happenings in the network based on the parameter values. Now, by changing the parameter values incrementally, one can observe the goodness in performance achieved and hence the process can be repeated until no further improvement is possible.

2.1.10.3 Performance Oriented System Design (POSD)

A constant looping process comprising of (i) measuring the performance and (ii) tuning the parameters, can help in improving the performance of a network considerably, however, it cannot be the remedial measure for a poorly designed network. Whole designing a network system, the designer is to consider that: (a) CPU speed is more important than network speed, (b) Reduce packet count to minimize software overhead, (c) Minimize context switches, (d) Minimize copying, (e) Should give more bandwidth but not lower delay, (f) Avoiding congestion is better than recovering from it and (g) Avoid timeouts.

2.1.10.4 Faster Processing of TPDU (FPTPD)

The processing overhead of TPDU (Transport Protocol Data Unit) can be characterized in terms of two basic components: (a) Overhead per TPDU and (b) Overhead per byte. A possible solution to fast TPDU processing is to separate out the normal case (one-way data transfer) and handle it specially. Once a sequence of special TPDUs is obtained to get into the ESTABLISHED state, TPDU processing becomes is straightforward until one side starts to close the connection.

2.1.10.5 Protocols for Gigabit Networks (PGN)

A major performance bottleneck in the current network scenario is due to rapid increase in communication speed against relatively lower CPU speed. In such scenario, a common assumption of the protocol designer that the time to use up the entire sequence space would significantly exceed the maximum packet lifetime, usually fails. Also, today the gigabit network designers should learn to design the network for speed, not for bandwidth optimization.

2.2 Part II: Anomalies in Network

Anomalies are instances (e.g., points, objects, events, patterns, vectors, samples, etc.) in data that do not conform with an acceptable notion of normal behaviour. Anomalies in a network also refer to circumstances[6] which cause the network operations to deviate from normal behaviour. Network anomalies may occur due to several reasons such as: overload in the network, malfunctioning of devices, network misconfiguration, malicious activities or network intrusions that interprets the normal network services. However, the anomalies in a network can be broadly categorized into two: (a) Network performance related anomalies and (b) Security related anomalies. Next, we discuss each of these two categories of network anomalies, their sources and causes in detail. Performance related anomalies may occur due to various network vulnerabilities.

2.2.1 Performance Related Network Anomalies

In a network system, vulnerabilities are inherent weaknesses in the design, implementation and management of the system, which render the system susceptible to threat(s). Existing known vulnerabilities are usually traced back to any of the following three sources:
(a) *Ill-design*: A common source of network vulnerabilities is due to the presence of flaws in the design of hardware and software. A common example of such a vulnerability is the *sendmail flaw* identified in the earlier versions of UNIX. It enabled the hackers to acquire privileged access to hosts.
(b) *Poor implementation*: Another important sources of network anomaly is

the poor or incorrect configuration of the system. Such vulnerabilities usually result due to lack of inexperience, insufficient training, or sloppy work. For example, a system configuration without having restricted-access privileges on critical executable files. It will allow an illegitimate user to tamper those files easily.

(c) *Lack of management*: Use of improper methods and inappropriate checks and balances are another sources of network vulnerabilities. Improper monitoring and lack of proper documentation can also provide scope of network vulnerabilities. For example, lack of guarantee that security steps are being followed and that no single individual can gain the total control of a system.

The following subsections describe the potential causes of network vulnerabilities[52]

2.2.1.1 Network Configuration Vulnerabilities

Vulnerabilities in the network also may be caused due to improper network configurations. There are five major configuration vulnerabilities as reported below.

(a) *Weak network architecture*: A network infrastructure environment may be developed and enhanced on demands, with very less emphasis on the potential security consequences of the changes. As a result, security hole/gap may come into existence over time, backdoors for the adversaries.

(b) *Lack of Data flow controls*: Absence or inappropriate use of data flow control, such as access control list (ACL) may provide scope for illegitimate users or systems to access the network.

(c) *Poor configuration of security equipment*: Use of default configuration of the security equipment such as router, etc. may lead to insecure open ports and the network services running on hosts also may easily lead to exploitation. Improperly configured firewall rules and router ACLs can allow illegitimate traffic.

(d) *Lack of backup of device configuration*: Due to non-availability of procedures for restoring network device configuration settings in the event of accidental or adversary-initiated configuration changes, it may be difficult to maintain system availability and to prevent loss of data. Backup procedures with well documentation should be available to keep the network up with

minimum failure time
(e) *Unprotected password transmission*: Unprotected password transmission over insecure channel are susceptible to eavesdropping by adversaries A successful disclosure of password will allow an intruder to disrupt security operations or to monitor network activity.
(f) *Use of password for longer period*: Use of password for longer period will allow the adversaries to make a successful offline dictionary attack. It will enable the adversaries to disrupt security operation.

2.2.1.2 Network Hardware Vulnerabilities

Lack of appropriate protection and control mechanism may lead to network hardware vulnerabilities. Following are the major causes of such vulnerabilities.
(a) *Inadequate physical protection of network equipment*: Inappropriate monitoring and controlling of the network equipment may lead to damage or destruction
(b) *Insecure physical ports.* Use of insecure USB (universal serial bus) and PS/2 ports could be another major cause of permitting unauthorized connections.
(c) *Lack of environment control*: Inappropriate or loss of environmental control could lead to overheating or melting of the processors and consequently may even shut down its operation to avoid damage to themselves.
(d) *Inadequate physical access control mechanism for equipment and network connections*: Non-restricted access and inappropriate use (or unauthorized manipulation of setting) of the network equipment may lead to theft/damage of sensitive data, software and hardware.
(e) *Lack of backup for critical networks*: Lack of periodic backup of critical network data may lead to sudden failure of the system.

2.2.1.3 Network Perimeter Vulnerabilities

Lack of security perimeter and appropriate control mechanism may lead to this special type of network vulnerability. Next we discuss three major causes of this type of vulnerability:
(a) *Lack of well defined security perimeter*: If the control network does not

have a well-defined security perimeter, it may be difficult to ensure the necessary security controls in the network. It could lead to unauthorized access to systems and data.
(b) *Improper configuration of firewall*: Improper configuration of firewalls could allow unauthorized or malicious data as well as software including attacks and malware to spread between networks, making sensitive data susceptible to monitoring/eavesdropping on the other network, and providing individuals with unauthorized access to systems.
(c) *Use of networks for non-control traffic*: The purpose and requirements of Control and non-control traffic are different, such as determinism and reliability, so having both types of traffic on a single network makes it more difficult to configure the network so that it meets the requirements of the control traffic.

2.2.1.4 Network Monitoring and Logging Vulnerabilities

Vulnerabilities in a network also may be caused due to improper logging and inadequate firewall setting. Appropriate and accurate logging can be of significant help to identify the causes of a security incident. A regular security monitoring will help to avoid occurrence of unnoticed incidents or sudden damage or disruption in the network.

2.2.1.5 Communication Vulnerabilities

This subsection presents four major types of communication vulnerabilities.
(a) *Lack of critical monitoring and control paths identification*: Unauthorized or illegitimate connections can leave a backdoor for attacks.
(b) *Use of well documented communication Protocol*: Adversaries can exploit protocol analyzer or other utilities for well documented and standard protocol to interpret the data sent by using protocols like File Transfer Protocol (FTP), telnet, Network File System (NFS), etc.
(c) *Lack of standard authentication of users, data or devices*: Due to lack of appropriate authentication, there could be chances to replay, manipulate, or spoof data or devices (e.g., sensors, user IDs, etc).
(d) *Lack of integrity checking*: Lack of integrity checks could enable the adversaries to manipulate communications undetected.

2.2.1.6 Wireless Connection Vulnerabilities

This type of network vulnerabilities may be caused due to two major reasons:
(a) *Inadequate authentication*: Use of a weak and biased mutual authentication between a wireless clients and an access point could lead to connect a legitimate client to a rogue access point of an adversary Also such a weak authentication may allow new adversary to gain access to the system.
(b) *Inadequate data protection*: Inadequate protection mechanism may allow the adversaries to gain access to the non-encrypted data

In the preceding subsections we have discussed the various possible sources of network vulnerabilities which may cause performance related anomalies in a network. Next, we discuss about the second category of network anomalies, i.e., security related network anomalies which is our prime concern

2.2.2 Security Related Network Anomalies

Anomalies are instances (e.g , points, objects, events, patterns, vectors. sample etc.) in data that do not comply with an acceptable notion of normal behaviour. Anomalies in network may occur due to several reasons (i) Network operation anomalies (ii) Flash crowd anomalies and (iii) Flood anomalies or malicious activities. Out of these occurrences of anomalies due to malicious activities is our main concern. Malicious activities in a network can be of various types such as point anomaly. contextual anomaly and collective anomaly^8. We state and illustrate each type of anomaly in terms of fraudulent credit card transactions.
Point anomaly: Point anomalies of an instance can be found exceptional or anomalous with respect to rest of data, such instance is referred point anomaly. For example, exceptional credit card expenditure in comparison to previous transactions Figure 2.17 shows an example of point anomaly In Figure 2 17, object O_1 is isolated from other group of objects C_1, C_2 and C_3. Object O_1 is a point anomaly

Contextual anomaly: With reference to a given context (e.g , minimum maximum range), if an instance can be found anomalous or exceptional, such instance is referred as contextual anomaly. For example, with reference to

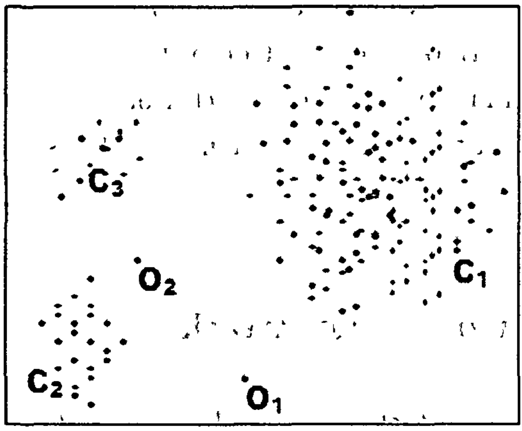


Figure 2.17: Point, collective and contextual anomaly

weekly credit card expenses of a Professor (other than festival expenditure), a credit card expenditure which significantly deviates from the usual range of weekly expenditure. An example is shown in Figure 2.17, where object O_2 is isolated in the context of group of objects C_2 Object O_2 is contextual anomaly.

Collective anomaly: With reference to a given normal behaviour, of a group of instances are found deviated or anomalous, such group of anomalous instances are referred as collective anomalies. In Figure 2.17, group of objects C_3 is distinctly separate from the groups C_1 and C_2. C_3 is a collective anomaly.

2.2.3 Who Attacks Networks

A computer network is a collection of interconnected systems running distributed applications. A network attack is a malicious attempt used to exploit the vulnerability of a computer or network attempting to break into or compromise security of the system. One who attempts to attack into a system is an attacker or intruder. Anderson[53] classifies attackers or intruders into two types: *external* and *internal*. *External intruders* are unauthorized users of the system or machines they attacked, whereas *internal intruders* have permission to access the system, but do not have privileges for the root or super user. *Internal intruders* are further divided into *masquerade intruders* and *clandestine intruder*. A *masquerade intruder* logs in as another user with legitimate access to sensitive-data whereas a *clandestine intruder*, the most dangerous, has the power to turn off audit control for themselves. Attack or intrusion is perpetrated by an inside or outside attacker of a system to gain unauthorized entry and control of the security mechanism of the system.

2.2.4 Precursors to an Attack

The precursors to an attack are basically a series of events used to trigger an attack. A network attacker executes a series of steps to achieve the desired goal. The order and duration of these steps is dependent on several factors including attacker's skill level, type of vulnerability to be exploited, prior knowledge information, and starting location of the attacker. The attacker's steps range from finding out the network via port scans, running exploits against the network, cleaning attack evidence and installing backdoors to guarantee easy access of the network later. The four basic actions taken by an attacker prior to execute a computer attack, as shown in Figure 2.18 are described next.

Prepare: In this step, the attacker attempts to collect network configuration information using port scanners to identify the vulnerability of the network. It tries to gather information such as computer IP address, operating systems, open ports with their associated listening software type and versions.

Exploit: Once the vulnerabilities are identified, the attacker attempts to exploit it during this step. The attacker may execute multiple number of attacks during this step.

Leave behind: After successful launching of the attack(s) via exploitation, the attacker often installs additional software to enable accessing the network later. Such 'leave behind' might be network sniffer or additional back-door network services.

Cleanup: Here, the attacker attempts to clean up any evidence left by the actions in the previous steps. This may include restarting daemons crashed during exploitation, cleaning logs and other information, and installing modified system software designed to hide the presence of other software from normal system commands.

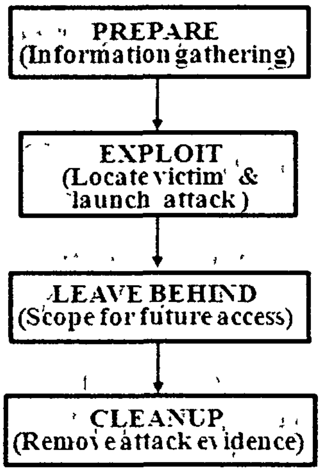


Figure 2.18: Basic steps of an attack execution

2.2.5 Network Attacks Taxonomy

An attack or intrusion is a sequence of operations that puts the security of a network or computer system at risk. Several network attack classification ways are available in literature[7,54,55,56]. A taxonomy or classification of attacks aims to provide a consistent way of specifying the relationships among the attacks, their classes and sub-classes. Attacks can be classified into *seven* main categories[7] based on their characteristics from implementation perspectives, as shown in Table 2.5. A brief discussion among these categories is reported next.

Infection: This category of attacks aims to infect the target system either by tampering or by installing evil files in the system.

Exploding: This category of attacks targets to explode or overflow the target system with bugs.

Probe: This category aims to gather information of the target system through tools. Co-ordinated probes or scanning are more serious attack of this category.

Cheat: A typical cheating characteristics of this category is their attempt to use fake or abnormality to the caller, for example, DNS (domain name server) spoofing, input parameter cheating etc.

Traverse: The common characteristics of this category is their attempt to crack the victim system through simple match by each possible key.

Concurrency: This category of attack targets to victimize (depletion) a sys-

Table 2.5: Taxonomy of Attacks

Main category	Sub-category
Infection	Viruses, Worms, Trojans
Exploding	Buffer Overflow
Probe	Sniffing, Port Mapping Security Scanning
Cheat	IP Spoofing, MAC Spoofing, DNS Spoofing, Session Hijacking, XSS (Cross Site Script) Attacks, Hidden Area Operation, and Input Parameter Cheating
Traverse	Brute Force, Dictionary Attacks, Doorknob Attacks, User to root (U2R), and Remote to local (R2L)
Concurrency	Flooding, DDoS (Distributed Denial of Service)
Others	All others

tem or a service by sending a mass of same requests which exceed the capacity that system or service could supply

Other: There are some attacks belonging to this special category, which attempts to infect the target system by using system bugs or weaknesses directly in attacks. Often, these may be found as trivial attempts, without requiring any professional skills. The network attacks can also be classified as passive and active. Passive attacks have an indirect effect. These are launched by the intruders for two major purposes: (i) to monitor and record traffic and (ii) to gather useful information for subsequent launching of an active attack. Due to their no overt activity, these attacks are not easily detectable. Packet sniffing, traffic monitoring and analysis are examples of passive attacks. On the other hand, active attacks are more dangerous. They have more overt actions on the network or system, which can be usually more devastating to a network. Active attacks are classified into four categories in DARPA [57] intrusion detection evaluation plan. The four categories of attacks are described in the following subsections.

2.2.5.1 Denial of service (DoS)

DoS is a very commonly found class of attacks with wide varieties, which attempts to block access to certain resources[58]. Typically, the access to those resources are blocked due to the inability of a particular network service such as e-mail, to be available or the temporary loss of all network connectivity and services. In general, a DoS attack is implemented with an objective to reset the target computer(s), or to consume its resources extensively so that no longer the intended service can be provided or blocking the access between the intended users and the victim so that no longer the communication can continue adequately between them. In some cases, this class of attacks attempts compromise millions of legitimate users for forced web sites access and compel to cease operation temporarily. Also, launching of such attacks successfully often can spread very fast across the branches of the network, and ultimately could lead to problems in the entire network. For example, by consuming the bandwidth of a router between a LAN and the Internet could lead to compromising not only the victim computer, but the entire network. Common forms of denial of service attacks are: *buffer overflow*, *ping of death (PoD)*, *TCP SYN*, *smurf*, *neptune*, *teardrop* and *land* in Table 3.2 of Chapter 3.

2.2.5.2 User to Root Attacks (U2R)

This class of attack is very difficult to distinguish from the normal traffic because it attempts to gain the system's superuser's privileges as one of the legitimate users. U2R attack initially attempts to access a normal user account either by sniffing passwords or by dictionary attack. Once it is successful to steal a legitimate user's password, it attempts to exploit the weaknesses of the system to gain privileges of the super user[58]. U2R attacks can be of various types: (i) *Buffer overflow* attack where, the attacker's program copies large volume of data into a static buffer without verifying the capacity of the buffer, whether it can hold or not. (ii) The other U2R attacks are like *rootkit*, *loadmodule*, *Perl*, etc. in Table 3.2 of Chapter 3.

2.2.5.3 Remote to Local (R2L)

The nature of this attack class is almost similar with U2R attacks. It attempts to send packets to a remote machine or host over a network without being a legitimate user of that machine or host. Then, either as a root or as a normal user, it tries to acquire access to the machine or host and execute malicious operations[59]. Two most common R2L attack examples are: *buffer* *overflow* and *unverified input* attacks. They are either done against public services (such as HTTP and FTP) or during the connection of protected services (such as POP and IMAP). Guessing password, for example, is an unauthorized access from a remote machine. The other common forms of R2L attacks are *warezclient*, *warezmaster*, *imap*, *ftp_write*, *multihop*, *phf*, *spy*, etc. in Table 3.2 of Chapter 3.

2.2.5.4 Probe

This class of attacks attempt to gain useful information about hosts, valid IP addresses, the services they offer, the operating systems used, etc. by scanning the network in various modes [60,59]. The attacker utilizes these information to identify the potential vulnerabilities of the system to launch an attack against selected machines and services. Some of the common types of this class of attacks are: (i) IPsweep, where the attacker scans the network for a service on a specific port of interest, (ii) portsweep, where the attacker scans through many ports to determine which services are supported on a single host, (iii) nmap, which is a tool for network mapping, etc. in Table 3.2 of Chapter 3. These scans are usually the precursor to other attacks.

A typical network is shown in Figure 2.19 with a protected LAN, DMZ (demilitarized zone) logical subnetwork and several attacker nodes and client nodes. In a DMZ (computing) subnetwork, additional security is provided from external threats. It consists of switches, several VLAN, nodes, servers and server for IDS.

2.2.6 Discussion

In this chapter, we introduce how computer networks work and how anomalies arise in networks. In the section of computer networks, we include a brief description of various media used in computer communications, reference models for layered architecture used in computer communications, protocols used in different layers of communication, various types of networks based on scale and requirement, different topologies of interconnections and physical components for network communications. In the section on anomalies in networks, we explain network anomalies in two major areas: performance related anomalies and security related anomalies. We describe many vulnerabilities may exist in

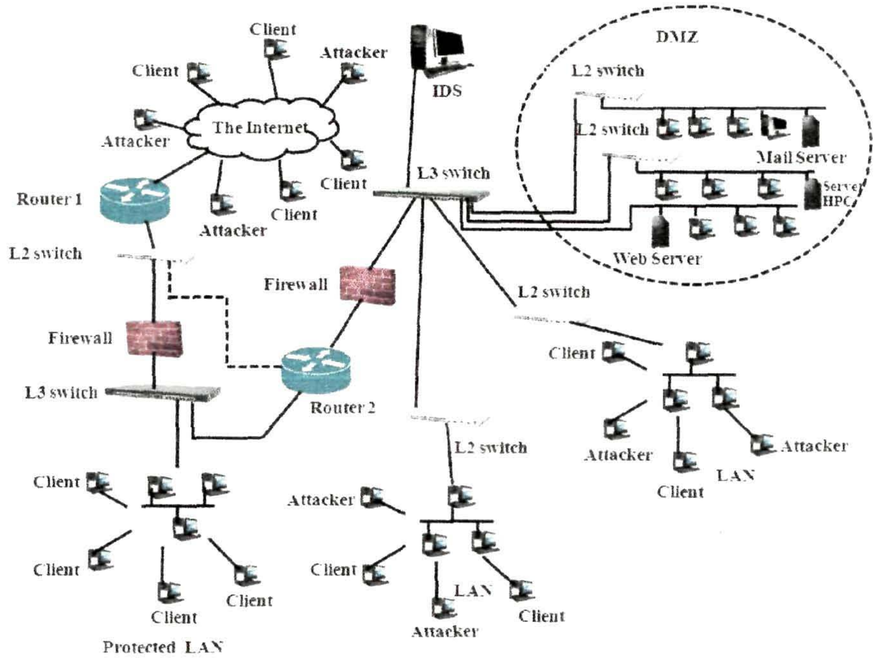


Figure 2.19: A typical view of a network with protected LAN, DMZ and attacking clients

networks allowing bad players to exploit them to create anomalies in performance related anomalies. In the security related anomalies part, we mention sources of anomalies in networks, types of network intruders, steps of network attacks (attack precursor) and types of attacks.

Several number of approaches are there for network anomaly detection. To test the performance of an approach, input data and evaluation metrics are unavoidable. We report various approaches of network anomaly detection methods, datasets and evaluation methods in the next *Chapter* 3.

Chapter 3

Detecting Anomaly in Network Data and Its Evaluation

Contents

In this chapter we introduce various data mining based network anomaly detection approaches, such as : *supervised*, *semisupervised*, *unsupervised* and *hybrid* and their architectures, components and various other important aspects. We also identify important issues for each of these architectures and analyze their capabilities and constraints or limitations in general. To evaluate an intrusion detection system, role of intrusion dataset is inevitable. In this chapter we discuss the features of some well known benchmark intrusion datasets, and also we include an elaborate discussion of the generation of such datasets. Finally, we discuss the various measures for evaluating the detection methods.

3.1 Detection of Network Anomalies

Anomaly detection is an essential multi step process for the mitigation of occurrences of anomalies or failures in time. It can compromise the defence mechanism, and degrade the performance of a network. Anomalies may be caused due to various reasons as discussed in the proceeding chapter. It may occur due to malicious actions (e.g., scanning, denial of service), or due to misconfigurations or failures of network hardware and software components (e.g., disruption of link, routing problems),. or even rightful events such as strangely large file transfers or flash crowds. The purpose of an anomaly detection mechanism is to analyze, understand and characterize network traffic behaviour, as well as to identify or classify the abnormal traffic instances such as malicious attempts from the normal instances. So, the anomaly detection problem also can be defined as a classification problem. Based on our survey[61] it has been observed that the existing anomaly classification methods work basically in three modes, as given below.

1. *Supervised anomaly detection*: Techniques trained in supervised mode assume the availability of a training data set which has labeled instances for normal as well as anomaly class. Typically, in such approach build a predictive model for normal versus anomaly classes. Any unseen data

instance is compared against the model to determine its belonging class. Two major issues arise in supervised anomaly detection. First, the anomalous instances are far fewer compared to the normal instances in the training data. Issues arising due to imbalanced class distributions are addressed in the data mining and machine learning literature[62]. Second, obtaining accurate and representative labels, especially for the anomaly class is usually challenging. A number of techniques have been proposed that inject artificial anomalies in a normal data set to obtain a labeled training data set[63].

Semi-Supervised anomaly detection: Techniques that operate in a semisupervised mode, assume that the training data has labeled instances for only the normal class. Since they do not require labels for the anomaly class, they are more widely applicable than supervised techniques. For example, in space craft fault detection[64], an anomaly scenario would signify an accident, which is not easy to model. The typical approach used in such techniques is to build a model for the class corresponding to normal behaviour, and use the model to identify anomalies in the test data.

2. *Unsupervised anomaly detection*: Techniques that operate in unsupervised mode do not require training data, and thus are most widely applicable. The techniques in this category make the implicit assumption that normal instances are far more frequent than anomalies in the test data. If this assumption is not true then such techniques suffer from high false alarm rate. Many semi-supervised techniques can be adapted to operate in an unsupervised mode by using a sample of the unlabeled data set as training data. Such adaptation assumes that the test data contains very few anomalies and the model learnt during training is robust to these few anomalies.

3. *Hybrid anomaly detection*: A hybrid intrusion detection method is developed by way of combining supervised and unsupervised network intrusion detection methods. The supervised methods have the advantage of known attack detection with higher attack detection accuracy rate and low false detection alarms. On the other hand an unsupervised method

has capability to novel attack detection. Thus; hybrid intrusion detection method is capable of detection of both known as well as unknown attacks with good attack detection accuracy rate.

Next, we define the anomaly detection problem for each of these approaches, discuss their generic architectures, components and various aspects in detail. Subsequently, we extend our discussion into three major categories. supervised, unsupervised and hybrid.

3.1.1 Supervised Anomaly Detection System

Supervised network anomaly detection basically attempts to classify network traffic into normal and different attack categories as early as possible based on the characteristics or features of the traffic with reference to a set of labeled training sample instances. So, the supervised anomaly classification problem can be defined as follows. For any given test instance $p_i(= x_1, x_2. x_3, .. , x_d) \in D_{test}$, i.e . a test dataset, for any given training instances $(q_1, q_2, \ldots, q_k) \in Q$, i.e , a training dataset, with class variable $c_i \in L$, i e., each training instance $q_i(= y_1, y_2, y_3, \ldots, y_d, c_i) \in Q$, the job of a supervised anomaly classifier is to identify the appropriate class label c_i for p_i as early as possible with high accuracy. Here, L can have two (binary class) or more (multi class) distinct values

A Generic Architecture

The responsibility of an intrusion detection system (IDS) is to (i) monitor the events occurring in a network or in a computer system, and (ii) to analyze them for the possible abnormal behaviour.

Anomaly based intrusion detection attempts to characterize the 'normal' behaviour of a system to be protected, and generate alarm information with a significant deviation observed due to an instance with reference to the predefined normal behaviours[8]. Another way is to define the role of an IDS is to model anomalous behaviour due to occurrence of certain event(s) and generates alarms when the deviation between the observed pattern and the expected pattern(s), becomes noticeable.

A generic architecture of supervised network anomaly detection system (supervised ANIDS) is shown in Figure 3.1.

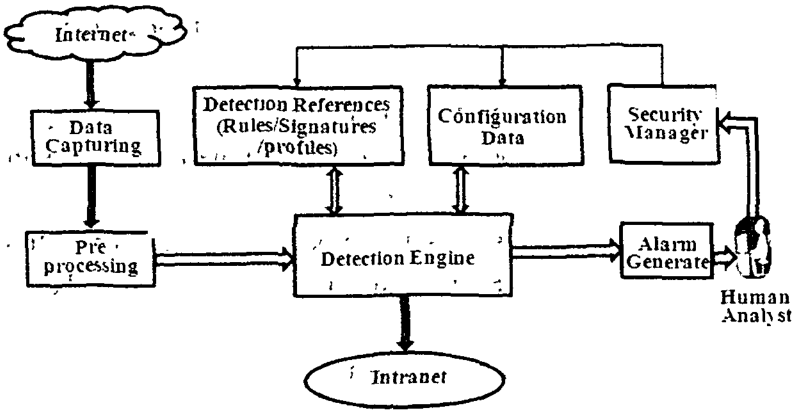


Figure 3.1: A generic architecture of supervised ANIDS

Basic Components

The generic model of a supervised ANIDS contains the following elements.

Data Capturing: Network traffic data is gathered using traffic capturing tool viz., *gulp* and stored.

Preprocessing: Raw stored data is filtered, important attributes are identified and features are selected.

Detection Engine: This is the central component of the anomaly detection system. It attempts to detect the anomaly using profiles of normal or anomalous behaviour. The model for identification of anomalies is created using *Reference Data* and *Configuration Data*.

Reference Data: It is the storage of the information about known intrusion signatures or rules or profiles of normal behaviour. In the later case the processing elements updates the profiles whenever new knowledge is acquired about certain observed patterns. This updates is performed in periodic intervals in a batch oriented fashion.

Security Manager: The role of a security manager (SM) is to update the signature database whenever knowledge about new type of intrusions become available. A novel intrusion analysis is a highly qualified task.

Configuration Data: It represents the intermediate results, e.g., partially fulfilled intrusion signatures frequently updated by the processing element. The needed space to store these active information can grow a bit large.

Alarm Generate: It handles all outputs from the system whether it is the most common one or that is an automatic resoonse to the susnicious activity.

Issues

In supervised anomaly detection, the normal behaviour model of system is established by training with labeled or purely normal dataset. These normal behavior models are used to classify new network connections. The system generates alert if a connection is classified to be abnormal behavior In practice, to train a supervised ANIDS, an exhaustive collection of labeled or purely normal data is not easily available. Acquiring such a complete and pure normal data is a time consuming task and it demands for the involvement of high skilled network security professionals.

3.1.2 Unsupervised Anomaly Detection System

Unsupervised anomaly detection can address the issue of novel intrusion detection without prior knowledge of intrusions or purely normal data. This approach does not require training data. In literature for unsupervised anomaly based intrusion detection[65,66,67,68], clustering is a widely found method. A clustering is a method of grouping of objects based on similarity of the objects. Clustering itself is a kind of unsupervised study method[69]. This method can be carried on unlabeled data, it divides the similar instances into the same group and divides the dissimilar instances into different groups. Unsupervised anomaly detection often tries to cluster test dataset into groups of similar instances which may be either intrusion or normal instances Although, using of clustering method in unsupervised anomaly detection for intrusion, generate many clusters, labeling of clusters is still a difficult issue faced by this approach. In order to label clusters, unsupervised anomaly detection approach models normal behaviour by using two assumptions[70]: (i) The number of normal instances vastly outnumber the number of anomalies and (ii) Anomalies themselves are qualitatively different from the normal instances. If these assumptions hold, intrusions can be detected based on cluster sizes. Larger clusters correspond to normal data, and smaller clusters correspond to intrusions. But this method is likely to produce higher false detection rate as the assumptions are not always true in practice.

A Generic Architecture

A generic architecture of unsupervised ANIDS is given in Figure 3.2.

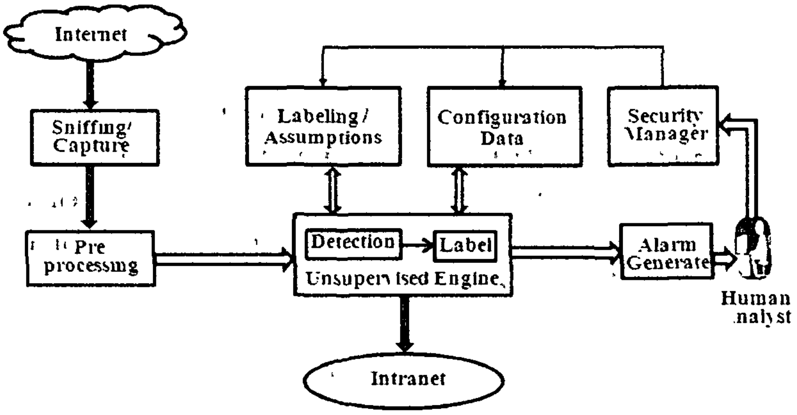


Figure 3.2: A generic architecture of unsupervised ANIDS

Basic Components

The generic model of a unsupervised ANIDS contains all those modules, as found in case of supervised ANIDS, except the Detection engine and the labeling technique. We discuss these two modules next.

Unsupervised Engine: This is the actual anomaly detection component of the system. It consists of two modules *Detection* and *Label*. Based on specific method or algorithm, *Detection* module perform detection either by grouping similar instances or by identifying exceptional instances in input data. *Label* module works after completion of *Detection* module to label the instances either as normal or anomalous. Anomaly detection model is created using *Labeling/Assumption* and *Configuration Data*.

Labeling/Assumptions: The usefulness of clustering has already been established in grouping the normal and malicious traffic instances in a network. It merely groups the data, without any interpretation of the nature of the groups. To support appropriate interpretation of those groups, labeling techniques are used. An example labeling strategy is available in[71], which attempts to solve this problem based on the physical characteristics (e.g., size, shape, compactness, etc.) of the clusters to interpret their nature. Cluster quality indexes are often found in use in distinguishing the clusters from each other. Dunn's index[72], C-index[73], Davies Bouldin's index [74], Silhouette index [75],and Xie-Beni index [76], are the indexes usually used for computing cluster validity.

Issues

Unsupervised anomaly detection approaches work without any training data or these models may be trained on unlabeled or unclassified data and they attempt to find intrusions lurking inside the data. The most prevalent advantage of the anomaly detection approach is the detection of unknown intrusions without any previous knowledge. In order to label clusters, an unsupervised ANIDS models normal behavior by using two assumptions[70]: (i) normal instances are significantly large in number than the anomalous instances and (ii) anomalies themselves are qualitatively different from normal instances. If these assumptions hold, intrusions can be detected based on size of cluster Largely populated clusters represent normal data, and smaller clusters correspond to intrusions. So, labeling of an instance is the vibrant issue of an unsupervised ANIDS.

3.1.3 Hybrid Anomaly Detection System

A hybrid anomaly detection approach combines both supervised and unsupervised methods of network anomaly detection. Supervised anomaly detection approaches can detect known type attacks very well but do not have capacity for unknown attack detection. On the other hand unsupervised methods can detect unknown attacks well. Typically, in a hybrid approach, advantages of both supervised and unsupervised approaches are combined for significant performance enhancement in anomaly detection. For instance, hybrids of supervised and unsupervised methods are used in[46,47,43,44]

A Generic Architecture

A generic architecture of hybrid ANIDS is given Figure 3.3. The modules belonging to this architecture are basically same with those found in case of supervised and unsupervised architecture..

Issues

The performance of an individual classifier either supervised or unsupervised is not equally good for classification of all categories of attack as well as normal instances. There is a possibility of obtaining good classification accuracy for all categories in a dataset by using an appropriate combination of multiple well performing classifiers. The objective of such a combination is to provide

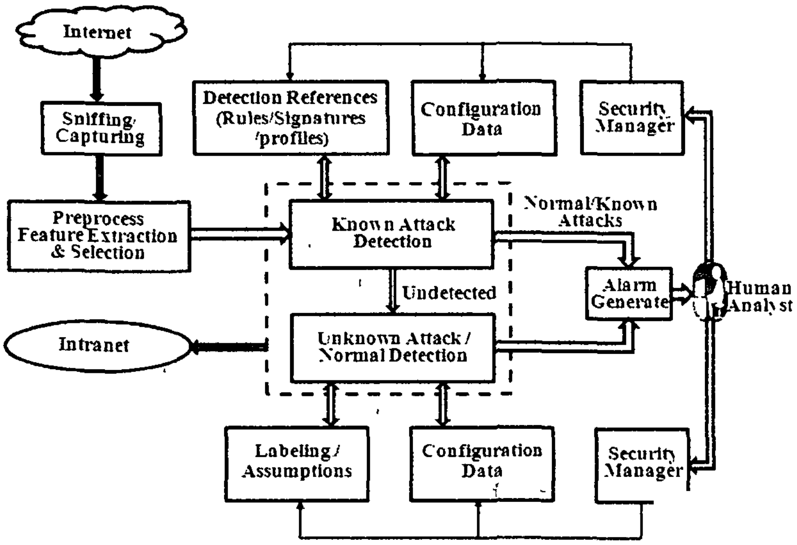


Figure 3.3: A generic architecture of hybrid ANIDS

the best performance from the participating classifiers for each of the classes of attacks. The selection of supervised or unsupervised classifier at a particular level for a given dataset is the critical issue for the hybrid ANIDS.

3.2 Aspects of Network Anomaly detection

3.2.1 Proximity Measure and Types of Data

In solving any pattern recognition problem such as classification, clustering or retrieval problem using supervised or unsupervised learning methods, proximity measure plays an important role. Proximity measure either can be a dissimilarity or a similarity measure. From mathematical point of view, distance, a synonym of dissimilarity, between a pair of objects is the quantity by which the objects are apart from each other. All distance measures may not satisfy the metric properties[77], i.e. *non-negativity, symmetricity* and *transitivity* properties. Those distance measures which do not satisfy the metric properties, are called *divergence*. Similarity measures are also referred as similarity coefficients. The choice of a proximity measure for any two objects depends on (i) type of data used to represent the objects, (ii) type of measurement required, and (iii) number of attributes or dimensionality. Based on the type of data used, i.e., *numeric, categorical* or *mixed-type*, proximity measures also differ. In order to define a distance measure between a pair of *numeric* data objects, which may be discrete or continuous, usually the geometric properties based on the representations of the objects are exploited. A detailed discussion on various numeric proximity measures is reported in [77]. To handle categorical data, either a data-driven categorical proximity measure can be used, or one can adopt appropriate procedure to encode the categorical data into binary or numeric data, and subsequently could use any proximity measure for numeric or binary data. A detailed discussion on categorical proximity measures is available in [78]. Handling of mixed type of data is rather more complex, especially when the attributes are with different weights. However, two straightforward ways of handling mixed type data are: (i) to represent categorical values using numeric values, and use numeric proximity measure, and (ii) convert the numeric values into categorical and use categorical proximity measures. Also, one can handle mixed-type data by directly finding proximity [79] between a pair of categorical values using exact matching approach. If the values are identical, distance will be 0, and 1, if they are distinct. A detailed discussion on mixed-type proximity measures is reported in [61].

3.2.2 Data Labels

The labels associated with a data instance denote if that instance is normal or anomalous. It should be noted that obtaining labeled data that is accurate as well as representative of all types of behaviours, is often prohibitively expensive. Labeling is often done manually by a human expert and hence requires substantial effort to obtain the labeled training data set. Typically, getting a labeled set of anomalous data instances which cover all possible type of anomalous behavior is more difficult than getting labels for normal behavior. Moreover, anomalous behavior is often dynamic in nature, e.g., new types of anomalies may arise, for which there is no labeled training data.

3.2.3 Relevant Feature Identification

Feature selection has taken important role in detecting network anomalies. Hence, feature selection methods have been used in the intrusion detection domain for eliminating unimportant or irrelevant features. Feature selection reduces computational complexity, removes information redundancy, increases the accuracy of the detection algorithm, facilitates data understanding and improves generalization. These algorithms have been classified into wrapper, filter, embedded and hybrid methods[80]. While *wrapper* methods try to optimize some predefined criteria with respect to the feature set as part of the selection process, *filter* methods rely on the general characteristics of the training data to select features that are independent of each other and are highly dependent on the output. Feature selection methods use a search algorithm to look up the whole feature space and evaluate possible subsets. To evaluate these subsets, they require a feature goodness measure that grades any subset of features In general, a feature is good if it is relevant to the output, but is not redundant with other relevant features. A feature goodness measure is always dependent one into another feature of the data.

3.2.4 Anomaly Score

Detection of anomalies depends on scoring techniques that assign an anomaly score to each instance in the test data depending on the degree to which that instance is considered an anomaly. Thus, the output of such a technique is a ranked list of anomalies. An analyst may choose to either analyze the top few anomalies or use a cut-off threshold to select anomalies. Several anomaly score estimation techniques have been developed in the past decades. Some of them have been represented under the category of *distance-based*, *density-based* and machine learning or soft computing based approach. A detail description of various anomaly scores are presented in[61].

3.2.5 Reporting of Anomalies

An important aspect for any anomaly detection technique is the manner in which the anomalies are reported. Typically, the outputs produced by anomaly detection techniques are *scores*. *Scoring techniques* assign anomaly score to each instance in the test data depending on the degree to which that instance is considered an anomaly. Thus, the output of such techniques is a ranked list of anomalies. An analyst may choose to either analyze top few

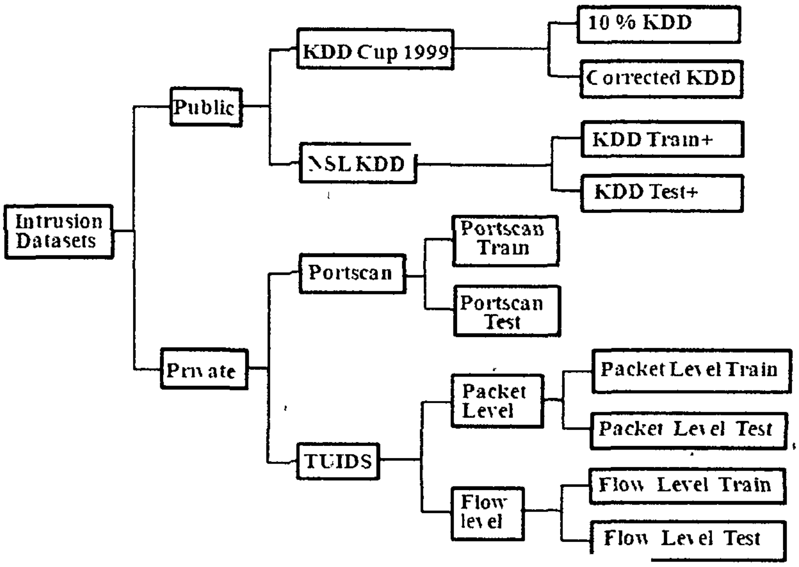


Figure 3.4: Classification of Intrusion Datasets

anomalies or use a cut-off threshold to select the anomalies.

3.3 Datasets

In the past couple of years, several intrusion datasets have been introduced by various security research groups to support evaluation of intrusion detection methods for known as well as unknown attacks (by introducing more number of attacks in the test datasets, while comparing to training datasets). In this section, we discuss some of such existing datasets available in the public domain. Apart from these datasets other private datasets generate as synthetically as well as by creating own testbed and by floating both normal as well as attack traffic (using appropriate launching tool for known attacks). So, the characteristics of these datasets are discussed under these broad categories (a) Public datasets (b) Synthetic dataset and (c) Real life private datasets. Out of these three, the synthetic datasets are not truly intrusion datasets, rather they are created to evaluate the outlier evaluation algorithm (discussed in detail in subsequent section) by considering all possible cases. A classification of datasets used in our work is reported in Figure 3.4.

Table 3.1: Attack distribution in KDD Cup datasets

Datasets	*DoS*	*U2R*	*R2L*	*Probe*	*Normal*	*Total*
Corrected KDD	229853	70	16347	4166	60593	311029
10-percent KDD	391458	52	1126	4107	97278	494021

3.3.1 Public Datasets

3.3.1.1 KDD Cup 1999 Dataset

The KDD Cup 1999[81] is an intrusion detection benchmark datasets. Here, according to network protocol, the connection between two network hosts is represented by each record in terms of 41 attributes (where 38 are combination of numeric continuous and numeric discrete and 3 are categorical attributes). Each record is labeled as either normal or one specific kind of attack. The attacks belongs to one of the four categories : Denial of Service (*DoS*), Remote to local (*R2L*), User to Root (*U2R*) and *Probe*. Number of samples of each category of attack in *Corrected KDD* dataset and 10 *percent KDD* dataset are shown in Table 3.1.

- Denial of Service(*DoS*) : Attacker attempts to prevent valid users from using a service of a system. For example, SYN flood, smurf, teardrop etc.
- Remote to Local (*R2L*) : Attackers try to gain entrance to victim machine without attaining an account on it For example, guessing password.
- User to Root (*U2R*) : Here, the attackers have access local victim machine and attempt to gain privilege of super user. For example, buffer overflow attacks.
- *Probe* : Attacker attempts to acquire information about the target host. For example, Port-scan, ping-sweep etc.

The description of attacks of each category of KDD Cup 1999 dataset is given in Table 3.2.

Table 3.2: KDD Cup Attack List

Name	Description
	***Category*: DoS**
Smurf	Smurf is a DoS attack where a spoofed source address is flooded with echo replies. The replies are caused when many ping (ICMP echo) requests using the spoofed source address are sent to one or more broadcast or multicast addresses.
Neptune	Here, session establishment packets are send against a victim machine using forged source IP address and resource of the victim machine is used up to wait for the session to be established.
Back	Back is another type of DoS attack where an attacker requests an Apache server with a URL consisting of several backslashes.
Teardrop	Teardrop is another type of DoS attack where it is attempted to reboot some systems by using misfragmented UDP packets.
Ping-of-death	In Ping-of-death (pod), large sized packets are sent over network by fragmenting and reassembling at destination using Ping command. On reassemble the excess packet size causes buffer overflow where the OS either crash, abort or hang.
Land	In Land attack, attacker attempts to modify the packet header by sending messages to a system with IP address indicating that the message is initiated from trusted host.
	***Category*: R2L**
FTP-write	The Ftp-write attack uses the advantage of a common anonymous ftp misconfiguration. If the anonymous ftp root directory or its subdirectories are owned by the ftp account or they are in the same group as the ftp account and they are not write protected, an attacker will be able to include files (viz., *.rhost* file) and successively obtain local access privilege to the system.
Guess-password	Guess-password attempts to discover password through telnet or guest account.
Imap	Remote buffer overflow uses the imap port leading to root shell.

Continued on next page

Table 3.2 – continued from previous page

Name	Description
Multihop	In Multihop, an attacker first breaks into one machine in a multi day scenario.
Phf	In Phf, exploitable CGI scripts are used to enable a client to execute any commands on a machine with a misconfigured web server.
Spy	In multi day scenario, an attacker breaks into one system for the purpose of finding sensitive information such that where the attacker has tendency to keep away from detection. The attacker applies several different exploiting methods to obtain entry to the system.
Warezclient	In Warezclient, an attacker attempts to download illegal software, previously posted by the warezmaster through anonymous FTP.
Warzmaster	Anonymous FTP upload of warez (Here a user attempts to log into an anonymous FTP site and execute a hidden directory to keep illegal copies of copyrighted software) onto a FTP server.
	***Category*: U2R**
Buffer-overflow	In Buffer-overflow, server receives data/commands from client and store in stacks (contiguous). An attacker wishing to infiltrate to the server send a block of data from a client longer than the application or process in expecting.
Loadmodule	Here, the attacker attempts to reset IFS for a legitimate user and a root shell is created.
Perl	Perl sets the user id to root in a perl script and creates a root shell.
Rootkit	In a multi day scenario, an attacker install one or more components of a rootkit.
	***Category*: Probe**
Ipsweep	It performs either a port sweep or ping on multiple host addresses.

Continued on next page

Table 3.2 – continued from previous page

Name	Description
Nmap	Network mapping using the nmap tool to map the network.
Portsweep	It performs a sweep across many ports to determine the services which are supported during a period on a single specific host.
SATAN	SATAN (Security Administrator Tool for Analyzing Networks), is a tool to probe the network for well-known vulnerabilities of a network.

3.3.1.2 NSL-KDD dataset

NSL-KDD[82] is a network-based intrusion dataset. It is a filtered version of KDD Cup 1999 intrusion detection benchmark dataset.. In the KDD Cup 1999 dataset, a large number of instances are redundant which can cause the learning mechanism biased towards those frequent records. To solve this issue, one copy of each record was kept in the NSL-KDD dataset. The NSL-KDD consists of two datasets: (i) $KDDTrain^+$, and (ii) $KDDTest^+$. The number of samples of each category of attack in NSL-KDD datasets are shown in Table 3.3.

Table 3.3: Attack distribution in NSL-KDD datasets

Datasets	***DoS***	***U2R***	***R2L***	***Probe***	***Normal***	***Total***
$KDDTrain^+$	45927	52	995	11656	67343	125973
$KDDTest^+$	7458	67	2887	2422	9710	22544

3.3.2 Private Datasets: Collection and Preparation

3.3.2.1 TUIDS Intrusion Dataset

The generation of TUIDS intrusion dataset involves a number of tasks to extract various types of features from network packet and flow features using an laboratory setup of isolated network. We use existing attack tools to generate a group of attacks against a local network server or host and collect the produced traffic as known attack traffic. The attacks for which we capture

Table 3.4: Attack List and the generating tools

Attack Values	**Generation Tool**	**Attack Values**	**Generation Tool**
bonk	*targa2.c*	1234	*targa2.c*
jolt	*targa2.c*	*saihyousen*	*targa2.c*
land	*targa2.c*	*oshare*	*targa2.c*
nestea	*targa2.c*	*window*	*targa2.c*
newtear	*targa2.c*	*syn*	*Nmap*
syndrop	*targa2.c*	*xmas*	*Nmap*
teardrop	*targa2.c*	*fraggle*	*fraggle.c*
winnuke	*targa2.c*	*smurf*	*smurf4.c*

data and the tools[83] used to generate the traffic are presented in Table 3.4. The description of the available tools used for generation of TUIDS are given in Table 3.5 and the attacks are given in Table 3.6. These tools and attacks are also used by Amini et al.[84].

Table 3.5: Tools for TUIDS Intrusion Data Generation

Tool	**Category, Description & Source**
Wireshark	It is a packet capturing tool. It is used for troubleshooting and analysis of network and development of software and communication protocol and for education. It uses cross-plateform *GTK+* widget toolkit to implement interface for its user, and using *pcap* for packet capture. It is comparable to *tcpdump*, but a graphical front end is its an added advantage with some integrated sorting and filtering options. It works in mirror port to extend capturing network traffic due to analysis for any tampering. Source: http://www.wireshark.org/

Continued on next page

Table 3.5 – continued from previous page

Tool	Category, Description & Source
Gulp	It is packet capturing tool. It has ability to read data directly from the network but is able to even-pipe output from legacy applications before writing to disk. *Gulp* allows much higher packet capture rate by dropping far fewer packets. If data rates increases, it realigns its writing to even-block boundaries for optimum writing efficiency. When it receives an interrupt, it stops filling its ring buffer but does not exit until it has finished writing whatever remains in the ring buffer. Source: http://staff.washington.edu/corey/gulp/.
tcptrace	It is a feature extraction tool. *tcptrace* takes input files produced by programs of capturing of packet (viz., *tcpdump*, *snoop*, *etherpeck*, HP Net Matrix, *Wireshark* and *Windump*. It produces several types of output information containing each connection seen (e.g., elapsed time, window advertisements, and throughput It can produce a number of graphs with packet statistics for further analysis. Source: http://jarok.cs.ohiou ed/software/tcptrace/.
nfdump	It is netflow data collection tool. It can collect and process netflow data on command line. It is limited only by disk space available for all the netflow data. It can be optimized in speed for efficient filtering. Filter rules are similar to the syntax of *tcpdump*. Source: http://nfdump.sourceforge.net/.
nfsen	It is netflow data collection & visualization tool. *nfsen* is a graphical web related front end for *nfdump* netflow tool It performs display of netflow data as Flows, Packet and Bytes using Round Robin Database (RRD). It can process the netflow data within the specified time frame. It can create history as well as continuous profiles. It can set alerts, based on various conditions. Source: http://nfsen.sourceforge.net/.

Continued on next page

Table 3.5 – continued from previous page

Tool	Category, Description & Source
nmap	*Nmap* (network mapper) is an attack launching tool (port scanning). It exploits the raw IP packets in a unique manner to examine what are the hosts available on the network, what services (application name and version) are offering by those hosts, what are running operating systems, type of firewall or packet filter used, etc. It is easy, flexible, powerful, well documented tool for discovering hosts in large network. It is free and open source tool. Source: http://nmap.org/.
rnmap	It is an attack launching (coordinated scanning) tool. *rnmap* (remote nmap) contains both client and server programs. Clients can connect to one centralized *rnmap* server and do their port scanning. Server does user authentication and uses *nmap* scanner to do actual scanning. Source: http://rnmap.sourceforge.net/.
Targa	Attack simulation tool. Targa is free and powerful attack generation tool. It enables to launch several attacks such as bonk, jolt, land,nestea, newtear, syndrop, teardrop, and winnuke into one multi-plateform *DoS* attack. Source· http://packetstormsecurity.nl/index.html.,

Table 3.6: Attack List

Name	Description
Bonk	*Bonk* manipulates fragment offset field in TCP/IP packets. By manipulating this number, it causes the attempted machine to reorganize a packet that is too large to be reassembled.
Jolt	*Jolt* communicated the target machine with a very large, fragmented ICMP packets. It fragments the ICMP packets in such a way to that the target machine is incompatible to reorganize them for use.

Continued on next page

Table 3.6 – continued from previous page

Name	Description
Land	This attack is launched by sending a TCP SYN spoofed packet with the IP address of the target host and an open port using as source and destination port both. *Land* makes the attempted machine to response to itself in continuous.
Nestea	*Nestea* launches IP fragments to a machine which is connected to the Internet or a network. *Nestea* is specific to the Linux operating system, and exploits a bug (commonly known as the 'off by one IP header' bug) in the Linux refragmentation code.
Newtear	*Newtear* attempts to exploit a problem with a smarter way. The Microsoft TCP/IP stack handles exceptions caused due to incorrect UDP header information, which changes padding length and increases the UDP header length field to twice the size of the packet.
Syndrop	When a system reconstructs a packet, it performs a loop to store it in a new buffer. Actually, there control the size of the packet only if it is too big. If the size of the packet is too small it can cause a kernel problem, which may lead to crash of the system
Teardrop	*Teardrop* exploits an overlapping IP fragment bug causes the TCP/IP fragmentation re-assembly code to improperly handle overlapping IP fragments. The fragmentation offset of the second segment is smaller than the size of the first and the offset plus the size of the second. This means that the second fragment contains the first.
Winnuke	*Winnuke* is a window based attack by sending OOB (Out-of-Band) data to an IP address of a Windows based machine connected to the Internet or network. This attack program connects through port 139, but other ports are also vulnerable if they are open. Upon receiving OOB data, the victim Windows machine cannot handle it and results with an exhibition of odd behaviour.

Continued on next page

Table 3.6 – continued from previous page

Name	Description
1234	This attack is launched by sending an oversize ping packet which cannot be handled by the network software. As a result the victim machine becomes very slow and ultimately it hangs. It may also result with loss of data.
Saihyousen	This attack is responsible for some firewalls to crash. It is launched by sending a stream of UDP packets. *Saihyousen* attack can cause consume all of the available resources and eventually cause a very messy reboot if the this occurs continuously for about 10-30 seconds after the froze of the machine.
Oshare	*Oshare* is a DoS attack. It is caused by sending a novel packet structure. The consequences of these attacks can different such as complete system crash, CPU load increasing, or momentary delays, depending upon configuration of computer. This will cause effect almost all versions of Windows 98 and NT-based systems with varying degrees related to the involved hardware.
Smurf	*Smurf* attack floods a system through ping messages using spoofed broadcast. It depends on a perpetrator by sending a large number of ICMP echo request (ping) traffic to IP broadcast addresses, where all of them have a spoofed source IP address of the targeted victim.
Fraggle	In *Fraggle*, a large number of UDP echo traffic are send by attacker to IP broadcast addresses, where all of them having a spoofed source address. This is similar to the smurf attack code rewrite.
Syn Scan	*SYN scan* attack is the default scan option. *SYN scan* acts against any compliant TCP stack. A SYN packet indicates the port is open (listening), while a RST (reset) is indicative of a non-listener. A port is marked as filtered if no response is received after several retransmissions. If an ICMP unreachable error is received, the port is also marked filtered . If a SYN packet is received in response, the port is considered open .

Continued on next page

Table 3.6 – continued from previous page

Name	Description
Xmass Tree Scan	The flags such as TCP FIN, PSH, and URG are set by *Xmass Tree Scan* attack. A port is considered closed, if a RST packet is received while it is open no response comes. If an ICMP unreachable error is received, the port is marked filtered.
Window Scan	When scanning reachable unfiltered systems, both open and closed port return a RST packet. *Window Scan* examines the returned RST packets of TCP Window field. On some systems, closed ports use a zero window size while a positive window size is used open port.. So instead of every time listing a port as unfiltered when it receives a RST return, if the TCP Window value in that reset is positive or zero, *Window scan* lists the port as open or closed, respectively.

Testbed Setup

The testbed setup of experiments for network traffic capture includes four L2 switches, one L3 switch, one router, three servers, two workstations and forty nodes. From the L3 switch and the L2 switches, six VLANs are created; and nodes and workstations are connected to separated VLANs. The L3 switch is connected to an internal IP router and the router is connected to the Internet through an external IP router To observe traffic activity to the switch, the server is connected to the L3 switch through a mirror port. Another two LANs of 350 nodes are connected to other LANs through two L2 switches and L3 switch and router. The attacks are launched within our testbed as well as from another LAN through the Internet. To launch attacks within the testbed, nodes of one VLAN are attacked from nodes of another VLAN as well as the same VLAN. Normal traffic is created within our testbed in a restricted manner after disconnecting the other LAN. Traffic activities to our testbed are observed on the computer connected to the mirror port. A diagram is shown in *Fig. 3.5* for generation of the TUIDS intrusion detection datasets using the testbed setup at *Network Security Laboratory* in the Department of Computer Science and Engineering of Tezpur University.

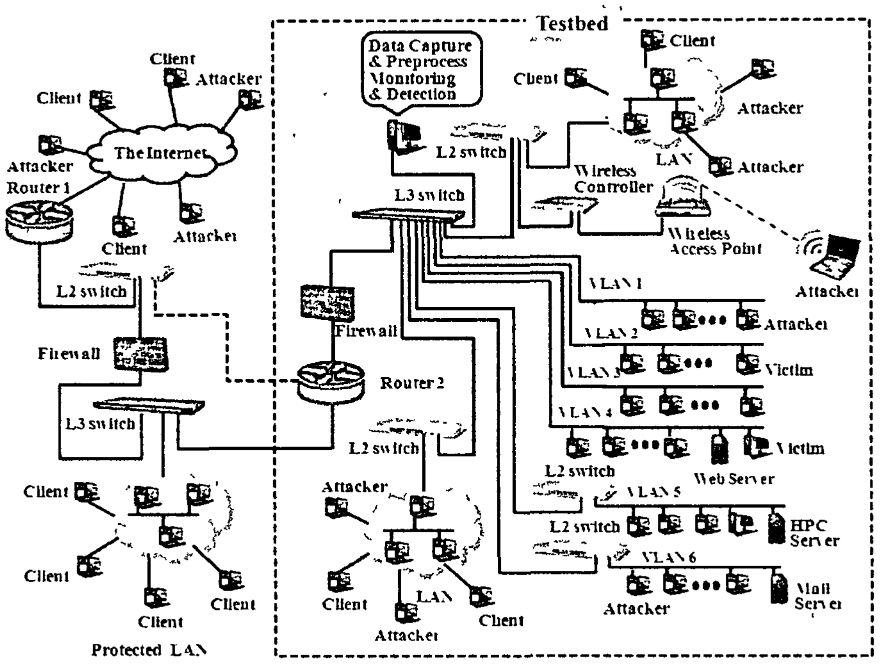


Figure 3.5: Testbed for generation of TUIDS Intrusion Dataset

Packet Network Traffic Feature Extraction

The packet level network traffic is captured using the open source software tool called *gulp*[85]. *Gulp* drop packets directly from the network and write to disk at high rate of packet capture. The packets are analyzed using the open source packet analyzing software *wireshark*[86]. The raw packet data is preprocessed and filtered before extracting and constructing new features In the packet level network traffic, 50 types of features are extracted. To extract these features we use open source tool *tcptrace*[87], *c* programs and *Perl* scripts. These features are classified as (i) *basic*, (ii) *content-based*, (iii) *time-based* and (iv) *connection-based*. The of list features are given in tables 3.7-3.10.

Network Flow Traffic Feature Extraction

The network flow data consists of a sequence of unidirectional packets passing through a point of observation in the network between source and destination hosts during a certain time interval. Belonging to a particular flow, all traffic has a set of common properties. The NetFlow protocol (IPFIX standard)[88,89] supports a summarization of the traffic to router or switch.

Table 3.7: Packet level Basic Features of TUIDS Intrusion dataset

Sl.	Feature Name	Type*	Feature Description
1	Duration	C	Time since occurrence of first frame
2.	Protocol	D	Protocol of layer 3- IP, TCP, UDP
3.	Src IP	C	Source IP address
4	Dst IP	C	Destination IP address
5.	Src port	C	Source port of machine
6.	Dst port	C	Destination port of machine
7.	Service	D	Network service on the destination e.g , http, telnet, etc
8.	num-bytes-src-dst	C	No. of data bytes flowing from src to dst
9.	num-bytes-dst-src	C	No. of data bytes flowing from dst to src
10.	Fr-no.	C	Frame number
11.	Fr-length	C	Length of the frame
12.	Cap-length	C	Captured frame length
13.	Head-len	C	Header length of the packet
14.	Frag-offset	D	Fragment offset value
15.	TTL	C	Time to live
16.	Seq-no.	C	Sequence number
17.	CWR	D	Congestion Window Record
18.	ECN	D	Explicit Congestion Notification
19.	URG	D	Urgent TCP flag
20.	ACK	D	Ack flag
21.	PSH	D	Push TCP flag
22.	RST	D	Reset RST flag
23.	SYN	D	Syn TCP flag
24.	FIN	D	Fin TCP flag
25.	Land	D	1 if connection is from/to the same host/port; 0 otherwise

Note- *(C-Continuous, D-Discrete)

Table 3.8: Packet level Content-based Features of TUIDS Intrusion dataset

Sl.	Feature Name	Type*	Feature Description
1.	Mss-src-dst-requested	C	Maximum segment size from src to dst requested
2.	Mss-dst-src-requested	C	Maximum segment size from dst to src requested
3.	Ttt-len-src-dst	C	Time to live length from src to dst
4.	Ttt-len-dst-src	C	Time to live length from dst to src
5.	Conn-status	C	Status of the connection (1-complete, 0-reset)

Note- *(C-Continuous, D-Discrete)

Table 3.9: Packet level Time-based Features of TUIDS Intrusion dataset

Sl.	Feature Name	Type*	Feature Description
1.	count-fr-dst	C	No. of frames received by unique dst in the last T sec from the same src
2.	count-fr-src	C	No. of frames received by unique src in the last T sec to the same dst
3.	count-serv-src	C	No. of frames from the src to the same dst port in the last T sec
4	count-serv-dst	C	No. of frames from dst to the same src port in the last T sec
5.	num-pushed-src-dst	C	No. of pushed pkts flowing from src to dst
6.	num-pushed-dst-src	C	No. of pushed pkts flowing from dst to src
7.	num-SYN-FIN-src-dst	C	No. of SYN/FIN pkts flowing from src to dst
8.	num-SYN-FIN-dst-src	C	No. of SYN/FIN pkts flowing from dst to src
9.	num-FIN-src-dst	C	No. of FIN pkts flowing from src to dst
10.	num-FIN-dst-src	C	No. of FIN pkts flowing from dst to src

Note- *(C-Continuous, D-Discrete)

Table 3.10: Packet level Connection-based Features of TUIDS Intrusion dataset

Sl.	Feature Name	Type*	Feature Description
1.	count-dst-conn	C	No. of frames to unique dst in the last N packets from the same src
2.	count-src-conn	C	No. of frames from unique src in the last N packets to the same dst
3.	count-serv-src-conn	C	No. of frames from the src to the same dst port in the last N packets
4.	count-serv-dst-conn	C	No. of frames from the dst to the same src port in the last N packets
5.	num-packets-src-dst	C	No. of packets flowing from src to dst
6.	num-packets-dst-src	C	No. of packets flowing from dst to src
7.	num-acks-src-dst	C	No. of ack packets flowing from src to dst
8.	num-acks-dst-src	C	No. of ack packets flowing from dst to src
9.	num-retransmit-src-dst	C	No. of retransmitted packets flowing from src to dst
10.	num-retransmit-dst-src	C	No. of retransmitted packets flowing from dst to src

Note- *(C-Continuous, D-Discrete)

Network flow is identified by IP addresses of source and destination as well as by port numbers. To identify flow uniquely, NetFlow also uses several fields, viz., the type of protocol, the type of service (ToS) from the IP header, and the router or the switch logical input interface. The router or the switch cache is used to store the flows and exported to a collector in case of the following constraints.

- Flows which have been idle for a specified duration of time are expired where default setting of specified time duration is 15 seconds, or the user can configure this time duration to be between 10 to 600 seconds.
- Flows which are lived longer than 30 minutes are expired.
- If the cache storage reaches its maximum size, a large number of expiry functions of heuristic nature are used to export flows.
- A TCP connection is finished using flag FIN or RST.

A flow collector tool, viz., *nfdump*[90] receives flow records from the flow exporter and stores them in a form suitable for further monitoring or analysis. A flow record is the information stored in the flow exporter cache. A flow exporter protocol defines how expired flows are transferred by the exporter to the collector. The information exported to the collector is referred to as flow record. NetFlow[91] version 5 is a simple protocol that exports flow records of fixed size (48 bytes in total).

Before analysis, all data is stored on disk. This partitions the process of storing data from analysis. In a time based fashion the data is organized. Nfdump has a daemon process *nfcapd* for flow record capturing which reads data from the network and stores them into files. Automatically, after every n minutes, typically 5 minutes, *nfcapd* rotates and renames each output file with the time stamp *nfcapd.YYYYMMddhhmm*. For instance, *nfcapd*.201012110845 contains data from December 11th 2010 08:45 onward. Based on a time interval of 5 minutes, this stores resulting in 288 files per day. The analysis of the data is performed by concatenating several files for a single run. The output is stored either in ASCII or in binary into a file and it is again ready to be processed with the same tool. We use c programs to filter and extract new features from the captured data. We remove

Table 3.11: Flow level Features of TUIDS Intrusion dataset

Sl.	Feature Name	Type*	Feature Description
	<u>**Basic features**</u>		
1.	Duration	C	Length of the flow (in sec)
2.	Protocol-type	D	Type of protocols- TCP, UDP, ICMP
3.	src IP	C	Src node IP address
4.	dst IP	C	Destination IP address
5.	src port	C	Source port
6.	dst port	C	Destination port
7.	ToS	D	Type of service
8.	URG	D	Urgent flag of TCP header
9.	ACK	D	Ack flag
10.	PSH	D	Push flag
11.	RST	D	Reset flag
12.	SYN	D	SYN flag
13.	FIN	D	FIN flag
14.	Source byte	C	No. of data bytes transferred from src IP addrs to dst IP addrs
15.	dst byte	C	No. of data bytes transferred from dst IP addrs to src IP addrs
16.	Land	D	1 if connection is from/to the same host/port; 0 otherwise
	<u>**Time-window features**</u>		
17.	count-dst	C	No. of flows to unique dst IP addr inside the network in the last T sec from the same src
18.	count-src	C	No. of flows from unique src IP addr inside the network in the last T sec to the same dst
19.	count-serv-src	C	No. of flows from the src IP to the same dst port in the last T sec
20.	count-serv-dst	C	No. of flows to the dst IP using same src port in the last T sec
	<u>**Connection-based features**</u>		
21.	count-dst-conn	C	No. of flows to unique dst IP addr in the last N flows from the same src
22.	count-src-conn	C	No. of flows from unique src IP addr in the last N flows to the same dst
23.	count-serv-src-conn	C	No. of flows from the src IP addr to the same dst port in the last N flows.
24.	count-serv-dst-conn	C	No. of flows to the dst IP addr to the same src port in the last N flows.

Note- *(C-Continuous, D-Discrete)

the unnecessary parameters and the retained parameters are *flow-start*, *duration*, *protocol*, *source-IP*, *source-port*, *destination-IP*, *destination-port*, *flags*, *ToS*, *bytes*, *packets-per-second (pps)*, *bits-per-second (bps)* and *bytes-per-packet (bps)*. Network traffic corresponding to attack and normal traffic is gathered using our local network within a 4 week period. A summary of the dataset is available in http://www.tezu.ernet.in/dkb[92]. 24 types of features are extracted and they are classified into three groups: (i) *basic*, (ii) *time-window based*, and (iii) *connection-based* features. The list of features is given in Table 3.11.

The network traffic data for attack and normal modes are captured using

Table 3.12: TUIDS Intrusion Datasets

Connection type	Dataset type			
	Training dataset		Testing dataset	
Packet level				
Normal	71785	58.87%	47895	55.52%
DoS	42592	34.93%	30613	35.49%
Probe	7550	6.19%	7757	8.99%
Total	121927		86265	
Flow level				
Normal	23120	43.75%	16770	41.17%
DoS	21441	40.57%	14475	35.54%
Probe	8282	15.67%	9480	23.28%
Total	52843		40725	

our local network. The attacks are generated using attack tools given in Table 3.4 against a server of the local network testbed. The created traffic in the process is collected and labeled as known attack traffic. There are generated 16 different types of attacks. The network traffic data was captured at packet level and flow level through two separate port mirroring machines. The captured data was preprocessed and filtered to extract various types of features. The numbers of records in the datasets are given in Table 3.12. We call the two datasets: *Packet Level* and *Flow Level TUIDS* datasets.

3.3.2.2 Portscan Dataset

The generation of *Portscan* dataset extracts various types of features from network packet data captured using an isolated network. The *Portscan* dataset is given in Table 3.13. The experimental testbed as shown in Fig. 3.5 was used for generation of *Portscan* dataset. With the help of existing attack tool, *Nmap*, we generate a number of attacks against the server of local network testbed and collect the generated traffic as known type attack traffic. Network traffic corresponding to normal and attack were captured through a port mirroring machine in our testbed in a three weeks period. The captured data were preprocessed and filtered. The extracted *Portscan* dataset features are given in Table 3.14.

Table 3.13: TUIDS Portscan Dataset

Connection type	Dataset type	
	PortscanTrain	PortscanTest
Normal	2445	1300
SYN	9750	2500
ACK	9945	4300
FIN	9780	3500
maimon	0	5145
null	0	9770
xmas	9740	3400
Total	41660	29915

Table 3.14: Features of TUIDS Portscan dataset

Sl.	Feature Name	Type*	Feature Description
1.	Duration	C	Time since occurrence of first frame (seconds)
2.	Frame length	C	Length of the frame
3.	Frame number	C	Frame number
4.	Capture length	C	Captured frame length
5.	TTL	C	Time to live
6.	Protocol	D	Protocol of layer 3- TCP
7.	Source IP	C	Source IP address
8.	Destination IP	C	Destination IP address
9.	Src port	C	Source port of machine
10.	dst port	C	Destination port of machine
11.	Length	C	No. of data bytes flowing
12.	Sequence number	C	Sequence number
13.	Header length	C	Header length of the packet
14.	CWR	D	Congestion Window Record
15.	ECN	D	Explicit Congestion Notification
16.	URG	D	Urgent TCP flag
17.	ACK	D	Ack flag
18.	PSH	D	Push TCP flag
19.	RST	D	Reset RST flag
20.	SYN	D	Syn TCP flag
21.	FIN	D	Fin TCP flag
22.	Window size	C	Window size
23.	Maximum segment size	C	Maximum segment size requested
24.	class ID		Class label

Note- *(C-Continuous, D-Discrete)

3.3.3 Synthetic Dataset

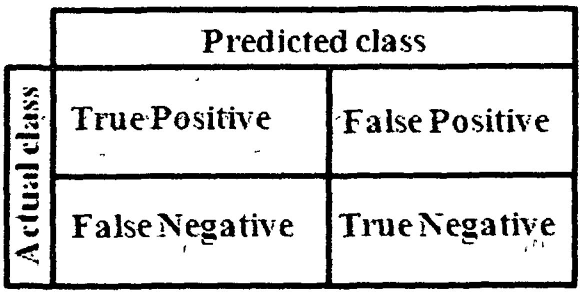
Synthetic data are generated to meet specific needs or certain conditions or tests that requirc to satisfy real data. This is usoful when dcsigning any system because the synthetic data may be used for simulation or for theoretical analysis so that the design can be modified for the requirement. This makes possible for finding a basic solution remedy, if the results prove to be unsatisfactory. As stated previously, synthetic data is used in testing and producing many different type of test scenarios. It allows the designer to build realistic behaviour profiles for normal users and attackers based on generated dataset to test a proposed method. In *subsection* 7.4.4, a two dimensional synthetic dataset is generated and used for testing a proposed method.

3.4 Evaluation Methods

As we are aware we cannot have anything which is totally or absolutely secure, without fear of compromise. Also, an evaluation or assessment of quality or accuracy of a system, mechanism or method is basically a snapshot in time. As time passes, the scenario or situation also changes, new vulnerabilities come up and accordingly the evaluation has to be redone, with new parameter tuning. However, it will be worth mentioning that the information obtained during an evaluation process has a significant role in the subsequent evaluation as well as in the final end product. In this chapter, we discuss 16 different evaluation measures under three broad categories: *accuracy*, *data* and *efficiency*.

3.4.1 Accuracy

To evaluate the performance of a ANIDS in terms of correctness, this metric is used. It measures the rate of detection and failure as well as the count of false alarms which is produced by the system[93,94]. A ANIDS with accuracy 95% implies that out of 100 instances it correctly classifies 95 instances in their actual class. Usually attacks occur in a diverse manner and the number of attack instances are generally smaller than the normal instances[70,62,95]. As a result, most ANIDSs generate large false alarms, which is not expected from an efficient intrusion detection system. This metric helps to evaluate a ANIDS,



	Predicted class	
Actual class	True Positive	False Positive
	False Negative	True Negative

Figure 3.6: Decision possibilities for 2-class problem

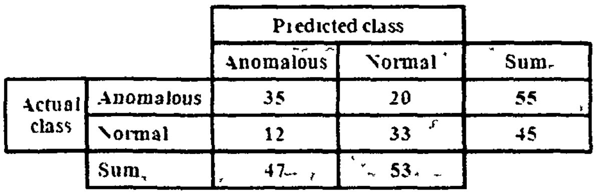
how correctly it can detect an attack. Accuracy of a ANIDS can be assessed in terms of five measures: (i) sensitivity and specificity, (ii) misclassification, (iii) confusion matrix, (iv) precision-recall and F measure and (v) ROC curve. In the following subsections each of these measures are discussed in some more detail.

3.4.1.1 Sensitivity and Specificity

To make an effective use of these two measures, a ANIDS developer models the network traffic classification problem as a 2-class (i.e. *normal* and *anomalous*) problem. It assumes the attack or anomalous data as positive, while the normal data as negative. Since during the classification of the traffic, outcomes can be right as well as wrong, it assumes *True* for right and *False* for wrong decision. As a results, as shown in Figure 3.6, there can be four possibilities out of these two variables: True Positive (*TP*), True Negative (*TP*), False Positive (*FP*) and False Negative (*FN*).

When a ANIDS correctly classifies an anomalous instance, we call it as *TP*, whereas an *FP* is said to occur when a legitimate action is misclassified as anomalous. Similarly, a *TN* occurs when a normal instance is classified correctly as legitimate action, while an *FN* causes when an anomalous instance does not detected by the ANIDS[96,62].

Sensitivity is the ratio between *TP* and (*TP*+*FN*) or in other words it is defined as (*TP*/(*TP*+*FN*)). Whereas, specificity is the ratio between *TN* and (*FP*+*TN*), or in other words, it is defined as (*TN*/(*FP*+*TN*)). Between these two measures, sensitivity can be set with high priority, when the system to be protected at all cost, and specificity gets more priority when efficiency is



		Predicted class		
		Anomalous	Normal	Sum
Actual class	Anomalous	35	20	55
	Normal	12	33	45
	Sum	47	53	

Figure 3.7: Classification of intrusion dataset

of major concern.

Example 1: Consider an intrusion dataset with 100 instances, out of which 55 are anomalous and 45 are normal instances. Now, assume that out of 55 anomalous instances, the ANIDS predicts 35 correctly (TP) and 20 as normal (FN); out of 45 normals, the ANIDS predicts 33 correctly (TN), and 12 as anomalous (FP). The classification of the intrusion dataset is shown in Figure 3.7. In this case, sensitivity and specificity will be (35/55) = 63.64% and (33/45) = 73.33%, respectively.

3.4.1.2 Misclassification Rate and PCC

In terms of network anomaly detection, misclassification is a situation where a ANIDS predicts a class (either normal or anomalous) that is different from the actual. This measure is found useful in estimating the probability of disagreement between the true and predicted statutes of a ANIDS by dividing the sum (*FN+FP*) by summed number of paired observations, i.e. (*TP+FP+FN+TN*). In other words, misclassification rate of a classifier can be defined as: (*FN+FP*)/*N*, *N=TP+FP+FN+TN*, total number of test class.

Alternatively. correct classification rate or *PCC* (percentage of correct classification) can be obtained by dividing the sum (*TP+TN*) by total number of paired observations, i.e. (*TP+FP+FN+TN*) as given below. In other words, *PCC* of a classifier can be defined as: (*TP+TN*)/(*TP+FP+FN+TN*).

For instance, the example shown in Figure 3.7, misclassification rate and *PCC* of the system are (20 + 12)/100 = 32% and (35 + 33)/100 = 68%, respectively.

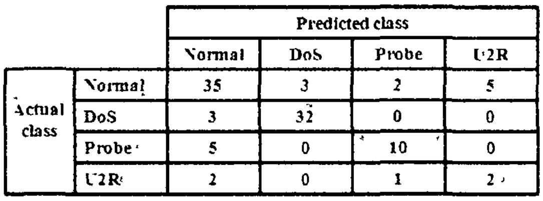
3.4.1.3 Recall, Precision and F-measure

Precision and recall are two well-known evaluation measures in the field of information retrieval. Precision is defined as the fraction of retrieved objects that are relevant to a given query or search request. Mathematically, it is the fraction obtained by dividing $|retrieved\ objects \cap relevant\ objects|$ by total retrieved objects, i.e., $|retrieved\ objects|$. Recall is the fraction of the objects that are relevant to a given query or search request that are correctly retrieved. Mathematically, it is the fraction obtained by dividing $retrieved\ objects \cap relevant\ objects$ by total relevant objects, i.e., $|total\ relevant\ objects|$. In case of a 2-class problem, recall is basically the sensitivity, or in other words, recall is the probability that a relevant object is retrieved by a search request or a query. Referring to 2-class problem (Figure 3.6), definitions for recall and precision can be given as: $(TP)/(TP+FN)$ and $(TP)/(TP+FP)$, respectively.

In case of network anomaly detection, precision measures the effectiveness of an ANIDS in identifying the anomalous or normal instances. A flagging is considered as correct and referred to as *TP* if the identified instance is actually from a malicious user. Very often, precision and recall are inversely proportional to each other and there is usually a trade-off between these two ratios. An algorithm which produces low recall and low precision is inefficient with conceptual errors most likely in the underlying theory. These types of anomalies or attacks that are not identified can suggest which part of the algorithm require more observation.

F-measure or balanced *F-score* is calculated by combining the *precision* and *recall*. F-measure (also known as $F1$ measure), usually is the harmonic mean of *precision* and *recall*[97], or mathematically, it can be defined as $2 \times (precision \times recall)(precision + recall)$. In case of an n-class intrusion classification problem, it is considered as a most preferable accuracy metric. $F1$ is maximum, when precision and recall both reach 100%, i.e., 1 means 0% false alarms in the classifier and detects all attacks, i.e., 100%. So, it is expected from a classifier to show $F1$-measure as high as possible.

Considering the example shown in Figure 3.7, the *Recall*, *Precision* and F-measure of the system are $(35/55) = 0.64$, $(35/47) = 0.75$, and $(2 \times 0.75 \times 0.64)/(0.75 + 0.64) = 0.69$, respectively.



		Predicted class			
		Normal	DoS	Probe	U2R
Actual class	Normal	35	3	2	5
	DoS	3	32	0	0
	Probe	5	0	10	0
	U2R	2	0	1	2

Figure 3.8: Confusion matrix for a 4-class problem

3.4.1.4 Confusion Matrix

It is a more general evaluation measure rather than those reported above for only 2-class problem. A confusion matrix can be used to evaluate the accuracy of a classifier for any n-class problem. Here, the size of the matrix depends on the existence of the number of distinct classes in the dataset to be detected. It helps to compare the class labels predicted by the classifier against the actual class labels. For better understanding, let us take the following example

Example 2: Consider an intrusion dataset with 100 instances, out of which 45 are normal, 35 are DoS, 15 are probe and 5 are U2R attack instances. Now, assume that out of 45 normals, the ANIDS predicts 35 correctly, and 5 as U2R, 2 as Probe and 3 as DoS; out of 35 DoS, the ANIDS predicts 32 correctly, and 3 as normal; out of 15 probe, the system predicts 10 correctly and rest 5 as normal; and finally, out of 5 U2R, the system predicts 2 correctly, and 2 as normal and 1 as probe. The confusion matrix for this situation is shown in Figure 3.8.

Here, the diagonal represents the correct classification of the ANIDS, and so, from the diagonal elements we can identify the incorrect predictions.

3.4.1.5 Receiver Operating Characteristics (ROC) Curves

The ROC curves are popular evaluation measure to visualize the relationship between True Positive (*TP*) and False Positive (*FP*) rates of an intrusion detection system. It has also an effective use in comparing the accuracy of two or more classifiers. However, the use of this measure is not restricted to only network traffic classification for anomaly identification. It was originated from signal processing theory and has applications over a large number of real fields such as bioinformatics, radiology, medical diagnosis as well as in artificial intelligence. It uses the orthogonal coordinate system[98] to observe the detection performance of a classifier, where the x-axis is used to represent the *FP*, while the *TP* is represented by y-axis as shown in Figure 3.9. Following conventions are used by an ROC curve to represent the accuracy of a classifier in the xy-plane[93,94,98,96].

1 The bottom-left point (0,0) is used to represent an ANIDS that performs with 100% normal identification accuracy, but zero false alarm rate. Such a ANIDS will be found capable of identifying all the time, all the data as normal, however, does not detect anything at the same time.

2. The top-right point (1,1) basically represents a ANIDS that eventually causes an alarm for each new traffic instance, it encounters. So, such a ANIDS will show 100% detection rate, however, at the same time also 100% false alarm rate.

3. Now, a line connecting these two points (mentioned above) represents any ANIDS that shows detection performance, which is basically a linear combination of those two points.The detection engine of such a ANIDS basically uses a randomized engine for detecting intrusions and so, in case of a real ANIDS, the ROC curves will always reside above this diagonal.

4. Again, the top-left point (0,1) means the performance of an ideal ANIDS with full detection rate, i.e., 100% and no false alarm, i.e., 0%. Hence, the accuracy of a ANIDS is considered to better, if it is closed to the ROC space.

3.4.2 Stability

For any network anomaly detection system, it is always expected to behave consistently in different network scenarios and for different circumstances. Also, the system should consistently generate similar alert messages for the identical events, so that the security manager can take necessary action without any ambiguity. Allowing the users to configure different alerts to provide different messages in different network environment may lead to an unstable state of the system.

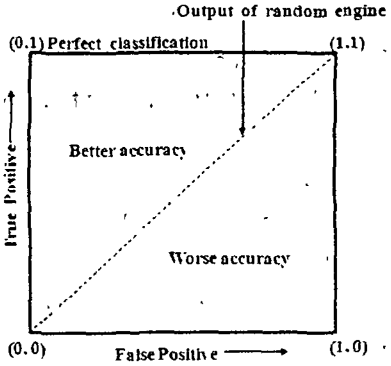


Figure 3.9: Receiver Operating Characteristics

3.4.3 Interoperability

While there's a deployment of multiple IDSs (may be ANIDS) in a same network, it can always be expected to have a time gap between the time of starting of an attack and the time of attack detection. An effective intrusion detection mechanism is supposed to be capable of correlating information from any of these sources or from a system log or a firewall log. This can help in maintaining interpretability, while deploying a range of IDSs/ANIDSs from various vendors. The effectiveness of an anomaly detection system is judged how correctly and fast (in real time), the system generates responses while attacks occur, However, having a good detection accuracy is no use if the detection time is too long and takes hours or days.

3.4.4 Data quality, validity and reliability

The usefulness of data for the purpose of intrusion detection depends on several factors such as source(s), correctness, timeliness, validity, reliability and consistency of data. Here we discuss the evaluation of an ANIDS based on data quality in terms three important parameters: quality, validity and reliability.

Quality of data is influenced by several factors, such as (i) source of data (should be from reliable and appropriate source), (ii) selection of sample (should be unbiased), (iii) sample size (neither over nor under sampled), (iv) time of data (should be recently updated real-time data) and (v) complexity of data (data should be simple enough to be handled easily).

Data validity implies whether the data used actually represent the same that we think is being measured. Though there are several types of validity, two types are most commonly accepted by most researchers, i.e., internal and external validity. An internal validity is to measure the strength of certainty that the observed effects in a detection experiment are the result of actual experimental treatment or operation (cause), rather than intervening, irrelevant extraneous or confounding variables. An appropriate use of the parameters or variables, the internal validity of data can be enhanced. External validity is more related with the result of research and their applicability (degree of use) in the real world. In other words, external validity attempts to measure the generic characteristics of a method or a classifier. However, attempting to increase the internal validity, may result in reduction of the degree of applicability of the research findings, i.e., the external validity. To identify the major factors influencing the data validity such as missing values, over-estimation/under-estimation or outliers, various techniques or statistical tests can be conducted by careful examination on the distribution patterns, frequency of occurrences and based on the values of the attributes.

Reliability implies that for any intrusion data used, the performances are complete (i.e., it includes all the variables and observations relevant to a given task), consistent (data should be unambiguous and clear enough, so that similar analysis will always give similar results), accurate (data should be originated from the actual or correct source(s) and also captured and preprocessed correctly) and purposeful (should serve the purpose meant for). To ensure the reliability of data, necessary intra- and inter-observer reliability analysis is to be carried out during capturing and collection of data.

3.4.5 Alert information

Alerts generated by a ANIDS should be meaningful enough to clearly identify (i) the reason causing the event to be raised, (ii) the reason the event is of interest and (iii) the source and target of an attack. It should assist the System Administrator or Analyst in determining the relevance and appropriate reaction to a particular alert. The appropriate source and the target information can help the analyst significantly in minimizing the damage to the systems or networks.

3.4.6 Unknown Attacks Detection

A vulnerability or attack outside the scope of definition of the existing profiles, is considered as an unknown attack. New vulnerabilities or exploits are evolving almost every day. Apart from the known attack detection, an anomaly based intrusion detection system should be capable of identifying unknown or modified intrusions. A ANIDS should be equipped with both supervised as well as unsupervised mechanisms to enable identifying both known and unknown or modified intrusions consistently.

3.4.7 Updating Profiles

A ANIDS should have provision to adapt with new vulnerabilities or attacks. Once new vulnerabilities or exploits are discovered, it should keep provision to update the profiles to enable identification of such vulnerabilities as known attacks in the future. However, considering the current high-speed network scenario, writing or modifying profiles in real-time without conflict with the existing rules or profiles is a challenging task.

In the subsequent chapter, we report a survey on existing intrusion detection methods, analyze their pros and cons and provide a general comparison awing to some of the well known detection methods.

CHAPTER 4

Network Anomaly Detection Methods and Systems

Contents

In this chapter we shall provide a comprehensive and structured survey on existing methods and systems for network anomaly detection. Based on the availability and use of labeled data, we discussed the general architectures of ANIDSes in the preceding chapter in three broad categories: supervised, unsupervised and hybrid. However, in this chapter we shall discuss the developments of ANIDSes in this evolving field of research and highlight the pros and cons of three reported methods in six distinct categories: supervised, unsupervised, probabilistic, soft computing, knowledge based and hybrid. A detailed discussion based on the comparisons among these anomaly based intrusion detection methods and systems also have been reported.

4.1 Network Anomaly Detection Method

A large number of methods have been introduced for identification of network anomalies based on monitoring and analysis of network traffic. Most existing methods attempt to identify anomalies by finding deviations from some underneath normal traffic model. Generally, models of these kind have to be trained with normal or attack free traffic traces for a longer duration of time. However, the training is practically a difficult problem. Also, it is difficult to clean the training data to be purely normal or 100% attack free. Further, this training process is to be repeated in periodic interval. Typically, network anomaly detection follows an approach of four stages. The network traffic data is collected and preprocessed in the *first* stage (data collection), while data analysis is done for extraction or selection of its relevant features (data analysis) in the *second* stage. *Third* stage is dedicated to monitor or analysis of the feature data; and the *fourth* stage performs the task of detection or classification of the traffic data over generally, a subspace (given by *second* stage) into normal or anomalous. A generic approach of this 4-step anomaly detection process is given in Figure 4.1.

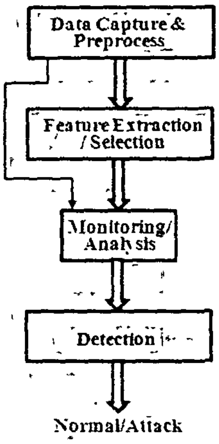


Figure 4.1: Basic steps ot anomaly detection

4.1.1 Requirements

Anomaly based network intrusion detection consists of analyzing and reporting unusual behavioural patterns in computing systems. It, typically, builds a normal system behaviour model from the observed data and characterizes any significant deviations or exceptions from this model. Followings are some important requirements for a ANIDS.

1. They should be capable of performing well, without prior knowledge of normal activities of the target system. Instead, they should have the ability to learn the expected behaviour of the system from observations.

2. The method should identify the malicious activities as accurately as possible for different network scenarios. In other words, the false alarm rate should be very negligible, if not zero.

3. Method should be least sensitive or dependent on input parameters.

4. Method should be capable of identify not only trivial attack types like isolated or bursty, but also any rare class or carefully launched attack.

5. Method should show near real time (if not true real time) detection performance with minimum false alarm rate.

6. Apart from known attacks, method should be capable of identify unknown or.novel attacks also.

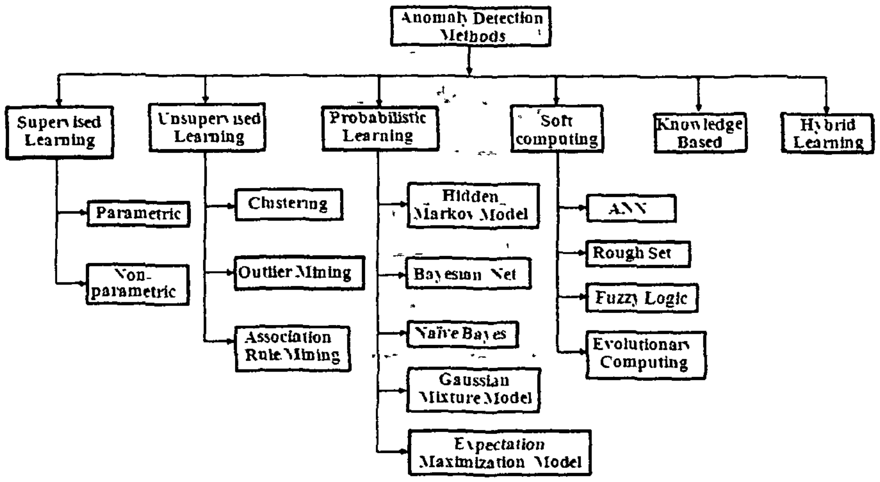


Figure 4.2: Classification of Anomaly based Intrusion Detection Methods

4.2 Types of Network Anomaly Detection Methods

Based on the requirements, detection mechanisms used, capability of detecting types of attacks, etc., the existing anomaly based network intrusion detection methods are categorized into six broad categories[99,100,101,102] as shown in Figure 4.2. Each type of methods have their own advantages and limitations.

4.3 Supervised Learning Methods

In supervised learning methods, it is assumed that the training dataset is available. The dataset has instances labeled as normal and anomaly classes. In such cases the typical approach is to build a normal versus anomaly behaviour class predictive model. New data instances are tested against this model to establish their belonging classes of normal or anomaly. Two major issues arise in anomaly detection using supervised approach. The first one is that in training data the number of normal instances are more compared to the anomalous instances. The second one is that obtaining of accurate and representative class labels, specifically for anomalous class is normally a critical issue. Large number of techniques[63] have been used which attempt to mix artificially generated anomalies in normal dataset for obtaining training dataset with labeled instances. Apart from these two issues, the building of a predictive model is the vital issue for supervised anomaly detection. In most cases, purely normal data or labeled data are not available readily. The generation of labeled data is related to consuming long time duration and very expensive for its manual classification. In practice, obtaining purely normal data is very hard as it is never been guaranteed to be no intrusion during the gathering of network traffic data. Supervised network anomaly detection methods are discussed in the following subsections.

4.3.1 Parametric Methods

In parametric methods, it is assumed that the normal data is produced by using a parametric distribution with parameters $\Im$ and probability density function (pdf) $f(p, \Im)$, where p is an instance of observation. The anomaly score of a test instance p is the inverse of the pdf, $f(p, \Im)$. The parameters $\Im$ are derived from the given data. A statistical hypothesis test [103], alternatively, can be used. H_0, the null hypothesis for such tests is the data instance p which has been generated using the required distribution with parameters $\Im$. Here, p is declared to be as an anomaly if H_0 is rejected by the statistical test. A statistical hypothesis test is closely related to a statistic test such that it can be used for obtaining a score of probabilistic anomaly for the data instance p. Many parametric anomaly detection methods are found in the literature[14,15,16].

Network intrusion detection deals with high-dimensional large volume network traffic data which is caused by the occurrence of high speed network traffic and large number of behaviour measures. Network intrusion detection requires to ensure an early detection of intrusion and generating alarm with a minimum delay for processing each event in a system to be secured. Therefore, for network intrusion detection, it is needed an anomaly based intrusion detection method with minimum computation cost and capability for handling multivariate data. In [14], on basis of a chi-square (χ^2) test, a distance measure is developed for obtaining a mean estimate of multivariate normal distribution as given in equation $\chi^2 = \sum_{i=1}^{n} \frac{(X_i - E_i)^2}{E_i}$.

Here, X_i is the i^{th} variable's observed value, E_i is the i^{th} variable's expected value and n is the total number of variables. If the expectation and observation of variables are close, the χ^2 will be small. Using $\bar{X}_1, \bar{X}_2, \ldots, \bar{X}_n$ as expected estimation, χ_i^2 is obtained as shown in equation $\chi^2 = \sum_{i=1}^{n} \frac{(X_i - \bar{X}_i)^2}{\bar{X}_i}$.

Here, χ^2 is the summation of observed and expected values squared differences for the variables and it has a normal distribution, approximately. For the χ^2 population, the mean and standard deviation can be computed by the sample mean $\bar{X}_2$ and the sample standard deviation S_X^2 from the sample data of χ^2. To detect anomalies, the in-control limits[104,105] can be set to 3 to attain 3-sigma control limits, $[\bar{X}^2 - 3S_X^2, \bar{X}^2 + 3S_X^2]$. Since the method is interested in detecting significantly large χ^2 values for intrusion detection, the upper control limit setting to $\bar{X}^2 + 3S_X^2$ is only needed. Altogether, for an observation if the computed χ^2 is greater than $X^2 + 3S_X^2$, it is signaled to be anomaly.

4.3.2 Non-parametric Methods

In non-parametric methods of anomaly detection, usually statistical models are used. Here, the structure of model is not defined a priori, rather it is determined based on given data. This class of methods does not assume an underlying model, rather, tailors its detection mechanism to the data Typically, such methods make less number of assumptions on distribution and characteristics of the data in comparison to parametric methods.

To maintain normal data profile, histograms approach has been found as the simplest and useful statistical non-parametric technique for intrusion detection[12,13]. Such methods are also known as frequency or counting based method. In these methods, certain profiles such as system, software or user, govern the behaviour of the data which can be handled efficiently with the use of histogram model. The basic histogram based method for multivariate data has been used to network anomaly detection by several authors[106,107]. In Packet Header Anomaly Detection (PHAD) and Application Layer Anomaly Detection (ALAD)[108], a variant of this simple technique is applied for network intrusion detection. The basic method is to construct histograms according to attributes for multivariate data. For a test instance during testing, an anomaly score is calculated for each attribute as its bin height. The anomaly scores for each attribute are summed for the test instance to obtain overall total anomaly score. It identify anomaly on basis of threshold anomaly score.

In[109], a non-parametric method of adaptive behaviour is proposed for anomaly detection. It is based on score functions which map sample data to the interval of $[0, 1]$. One such score function is obtained from a K-nearest neighbour graph (K-NNG). Here, an anomaly is considered whenever a test sample score falls below a predetermined error δ of false alarm. Let $Q = \{q_1, q_2, \ldots, q_m\}$ is the training set of size m which belongs to $[0, 1]^d$, unit cube . q_{m+1} denotes a test point. The task is to identify the test point whether it is nominal data consistently or it is deviation from nominal data. If the concerned test point is an anomalous point then it is considered to come from a combination of underlying nominal distribution and a different known density in the training data The K-NNG or equivalently ε-neighbour graph (ε-NG) is constructed with the use of a distance function. K-NNG is constructed by connecting every q_i to the K number of closest points $\{q_{i_1}, q_{i_2}, \ldots, q_{i_K}\}$ in $Q - \{q_i\}$. The K nearest distances are sorted for every q_i in ascending order $d_{i,i_1} \leq \ldots \leq d_{i,i_K}$ and indicate $R_Q(q_i) = d_{i,i_K}$, which represents distance between q_i and its K-th nearest neighbour. ε-NG is constructed where q_i and q_j are connected if and only if $d_{ij} \leq \varepsilon$. In this case $N_Q(q_i)$ is defined as the rank of point q_i in the ε-NG.

In ordinary case if anomalous density is an arbitrary mixture of nominal and uniform density, the Equations (4.1) and (4.2) are considered associated score functions for K-NNG and ε-NG graphs ,respectively. These score functions map γ, a test data to the interval of $[0, 1]$.

$$K\text{-}LPE : \hat{p}K(\gamma) = \frac{1}{n} \sum_{i=1}^{m} \coprod \{R_Q(\gamma) \leq R_Q(q_i)\}, \tag{4.1}$$

$$\varepsilon\text{-}LPE : \hat{p}\varepsilon(\gamma) = \frac{1}{m} \sum_{i=1}^{m} \coprod \{N_Q(\gamma) \leq N_Q(q_i)\}, \tag{4.2}$$

where $\coprod(.)$ defines an indicator function. *K-LPE* (or *ε-LPE*), the score functions measure the relative concentration of point γ in comparison to the training set.

Finally, for a given pre-determined significance level β (e.g., 0.05), γ is identified as anomalous if $\hat{p}K(\gamma), \hat{p}\varepsilon(\gamma) \leq \beta$.

4.4 Unsupervised Learning Methods

Supervised methods require labeled data. An accurate labeled data which can represent all types of behaviour is often expensive for its prohibitive cost In usual process, labeling is done manually by a skilled human expert and thus, it entails substantial time and effort to acquire the labeled data. Generally, a labeled dataset containing data instances with labels as normal behaviour can be obtained more easier way than a labeled dataset of instances labeled as anomalous which can cover all possibly available types of anomalies. Besides, anomalous behaviour is of course dynamic in its nature. Novel types of anomalous behaviour may occur which may not have any labeled training data. On the other hand, no training data is needed in unsupervised anomaly detection approaches[110] as stated in *Chapter 3* and hence, this type of anomaly detection method are widely applicable. An unlabeled dataset is taken as input for such an approach and they attempt to seek intrusion instances lurked inside the data. These detected intrusion instances then can be used for training of misuse based detection methods or supervised methods of anomaly detection. The unsupervised method of anomaly detection is discussed in the following three subsections.

4.4.1 Clustering Methods

Clustering methods group data into clusters based on a provided similarity measure or distance computation from some reference points. The most commonly used procedure[70] for clustering begins with selection of representative points for each cluster. According to proximity to a corresponding representative point, each test data point is grouped as belonging to the cluster created with the representative point. The clustering has ability to learn and detect anomalies without requiring explicit explanations of classes or types of anomalies from system administrator in test data. Consequently, the requirement of training data is nullified for anomaly detection based on clustering method. Clustering has wide application in[17,18,19] for network anomaly detection.

Based on clustering and classification, an anomaly detection approach is presented by Yang et al in[19] for detection of intrusion. It is given in Figure 4.3. In this approach, clustering is done for the training data points to group

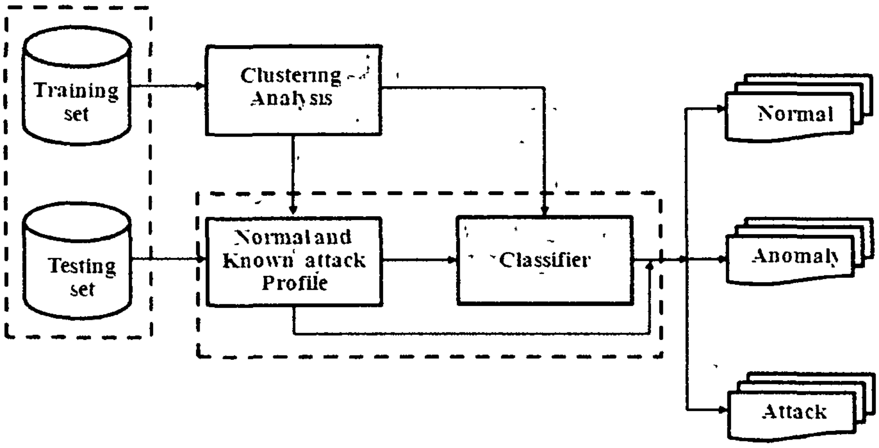


Figure 4.3· Clustering based Anomaly Detection

into clusters such that 'to select some of clusters as well-known attacks and normals on basis of fulfilling certain criteria. Thus, profiles are made for two classes. The training data points excluded from the profile are used to build a specific classifier. During the testing stage, they utilize an influence-based classification algorithm to classify network behaviours.

In[111], Casas et al. introduce a knowledge independent, anomalous network traffic detection approach referred as unsupervised network anomaly detection algorithm (UNADA). UNADA uses a novel clustering method, on basis of subspace-density clustering to obtain clusters and to recognize the outliers. The clusters are created in multiple numbers and with lower dimensions from the dataset. On basis of the traffic structure given by the multiple clusterings, they are combined: Thus, an abnormality ranking traffic flows are produced on basis of a distance based approach of correlation from the combined multiple clusterings. The working of UNADA is as given in Figure 4.4. Here, x_i represents subspace of features and P_i represents partition of $i = 1, 2, \ldots, n$ numbers.

UNADA performs unsupervised anomaly detection. The traffic capturing is performed in fixed length consecutive time slots and are aggregated into IP flows. IP flows are additionally aggregated at different flow levels using different aggregation keys. Thus, there is coarse to fine-grained resolution. To detect anomalous time slots, time series are built with traffic metrics including IP flows per time slot, the number of bytes and packets. Aggregation keys are

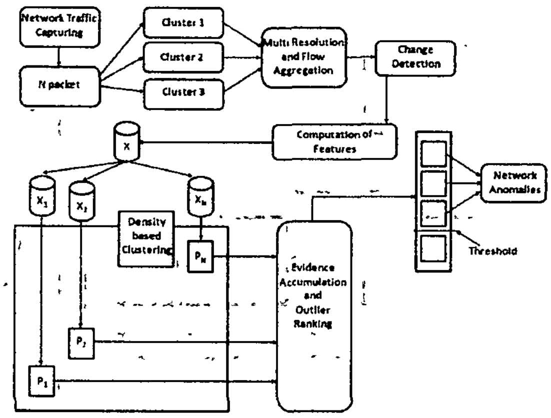


Figure 4.4: Working of UNADA

used for the purpose. A change detection method is then used on ,the time series, such that at the arrival of every new time slot, the change detection method analyzes the time series graphs,,using each aggregation key.

IP flows in the flagged time slot are used as the unsupervised attack detection. At this step, UNADA works for ranking of the degree of abnormality for every flow by using analysis of created clusters and outliers. Thus, at two different resolutions either IP source or IP destination, the aggregation key IP flows are analyzed. There are two different anomalies on the basis of which traffic anomalies can be classified, anomalies of 1-to-N or anomalies of N-to-1. When many IP flows are transferred from the same source to different destinations they are said to be anomalies of 1-to-N. Likewise, N-to-1 means IP flows from different sources to one destination. Anomalies of 1-to-N are highlighted by IP source, while anomalies of N-to-1 are detected more easily with IP destination key. Even when there are highly distributed anomalies, the use of both keys, i.e., IP destination key and IP source key number of IP flows, can be used to find outliers. The unsupervised network attack detection algorithm is based on clustering. Homogeneous groups of similar characteristics or clusters are formed by partitioning a set of unlabeled samples. Outliers are those samples that do not belong to any of these clusters. It is important to identify the cluster properly to determine the outliers. The aim is to determine or rank how different these are. Using a simple threshold detection approach, outlying flows which are top ranked are flagged as anomalies.

4.4.2 Outlier Mining Methods

Outlier mining methods search for detection of objects which do not match rules and expectations suited for majority of data. In view of clustering algorithm, the outlier objects in a data set are outside objects of the clusters. Thus, with reference to anomaly detection, the outliers may be considered as attacks. The researchers have been studying the outlier concept as a different useful domain area[103,61]. The detection of outliers often depends on the used methods and considered assumptions with respect to used data structures. Based on the approaches applied in detection of outlier, the methods can be classified[61] as distance-based, density-based and other methods. The distances among the objects in a dataset are computed with clear geometric interpretation in the distance-based outlier detection methods. In density-based outlier detection method, the density of neighborhood objects for each data instance are estimated. An object lying in a low density neighborhood is considered as an outlier. On the other hand, an object that lying in a high density neighborhood is considered to be normal. A number of works have been proposed in the domain of outlier detection with the objective of network anomaly detection[112,113,79].

In LOADED[113], Ghoting et al. introduce a distance based outlier detection method for mixed attribute data. Here, data points are considered linked if they are similar to each other for each attribute pair. Breunig et al. [114] compute a local outlier factor (LOF) for the objects in the dataset. The outlier factor quantifies outlyingness of an object. In ODMAD[79], computation of anomaly score is done for the data point considering the anomalies of the continuous values, the categorical values, and the mapping between the two spaces in the dataset. The data points are more likely outliers which have similarity near to '0'.

Minnesota Intrusion Detection System (MINDS) [112] uses data mining techniques for detecting network anomaly. The architecture of MINDS is shown in Figure 4.5. In the first step, MINDS extracts relevant important features which are used in detection method. Then, it summarizes the features based on time-window. After the feature identification step, the module of known attack detection is used for detection of network connections for which attack signatures are available. Thus, these connections are removed from further

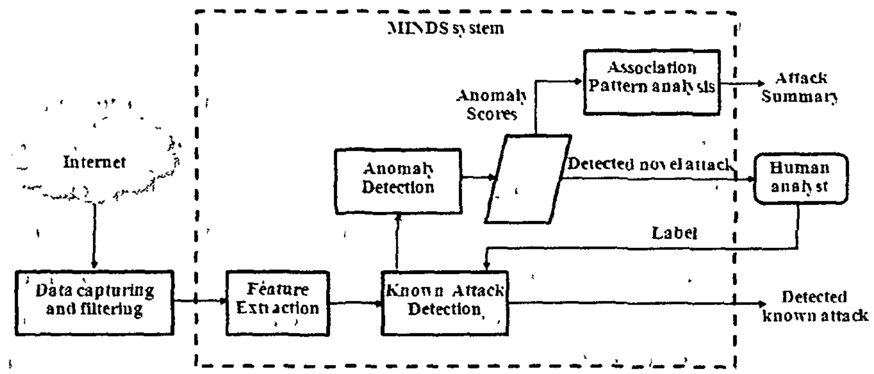


Figure 4.5: Architecture of MINDS

analysis. There after, the data is transferred to the anomaly detection module. A *quantitative measure of outlying, (LOF)* [114] *is computed for each object* by the anomaly detection module. The LOF considers the density of the neighbourhood objects around the observing point for determining its outlierness. The objects with higher LOF values are considered as outlier. The work of a human analyst here is to determine in case of the most anomalous connections whether they are truly attacks or interesting behaviour After analyzing the created summaries the analyst provides feedback to decide for using the summaries in creation of new rules to be used for known attack detection.

4.4.3 Association Rule Mining Methods

Association rule mining (ARM)[115,116,117] is a data mining method. This method describes events which are intended to occur together. The association rules can be described as follows: Given a transaction database D such that each transaction $P \in D$ represents a set of items in the database, an association rule is an implication of the form of $A \rightarrow B$ where $A \in D$ and $B \in D$ are sets of attribute-values, with $A \bigcap B = \emptyset$ and $\|B\| = 1$. The set A is the antecedent of the rule while the item B is the consequent. Two parameters are associated with the rule: support and confidence. The rule $A \rightarrow B$ has support m in the transaction set P if $m\%$ of transactions in P contain $A \bigcup B$. The rule $A \rightarrow B$ has confidence t if $t\%$ of transactions in P that contain A also contain B. Association rule mining approaches have been used to find normal patterns for anomaly detection[20,21,22,23]. Particularly, these methods

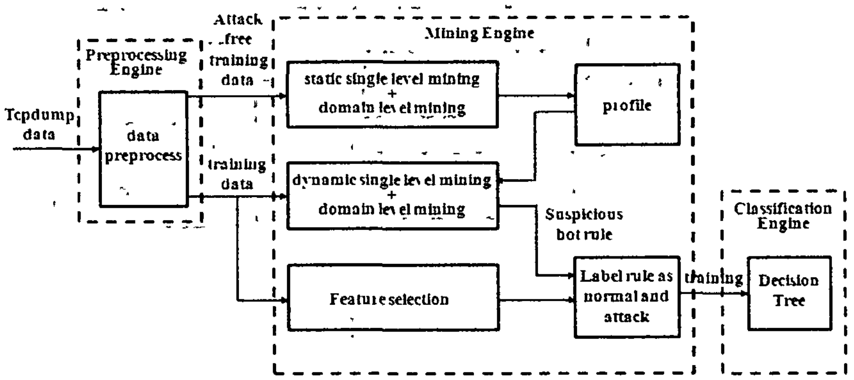


Figure 4.6: Training Phase of ADAM

are important in anomaly detection since there is scope of using association rules in constructing summary of anomalous connections which are detected by a system.

Daniel Barbara et al. propose an ARM based anomaly detection method, ADAM (Audit Data Analysis and Mining)[22]. A combination of ARM and classification are used in ADAM for identifying attacks in a TCPdump audit data. Initially, ADAM builds a 'normal' frequent itemsets repository which are logged during attack-free periods. This is performed by mining the data which is free of attacks. Next, a sliding-window algorithm is run to find frequent itemsets from the connections and comparisons are done with the stored repository of normal itemset. It discards the likely normal one. A previously trained classifier is used by ADAM for classifying the suspicious new connections to be as known attack, unknown type or as false alarm. ADAM is consists of three modules: preprocessing engine, mining engine and classification engine. The work of preprocessing engine is to sniff TCP/IP traffic data, and extraction of header information from each connection based on a predefined schema. The mining engine works for mining association rules for the connection records. It performs in two distinct modes: training (as shown in the Figure 4.6) and detection (as shown in the Figure 4.7). In training, a profile of the users and the system's normal behaviours is build and labeled association rules are generated, which are utilized to train the classification engine. In the detection mode, the mining of the unexpected association rules are done which are different from the profile. These unexpected association

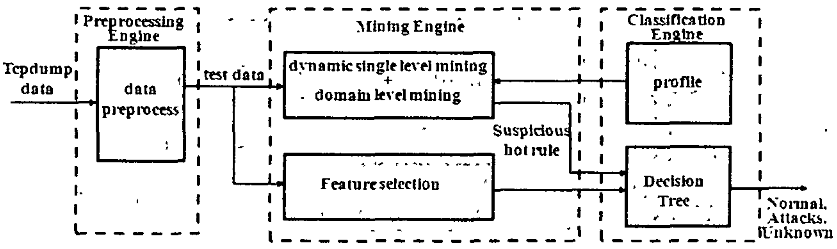


Figure 4.7: Detecting mode of ADAM

rules are classified by the classification engine into normal events and abnormal events. Abnormal events are classified further for their attack names.

4.5 Probabilistic Learning Methods

Probabilistic learning methods provide the mechanism to enable us to evaluate the outcome of system's performance affected by probabilistic uncertainty or randomness: For example, Bayesian Belief Networks which is based on the original work of Bayes[34] and Dempster-Shafer's theory of belief that is developed independently by Dempster[118] and Shafer[119] give us the technique of probabilistic learning method. The important characteristic of probabilistic learning is its capability to restructure them with recent available evidence to update previous performance of outcome.

4.5.1 Hidden Markov Model

The Hidden Markov Models (HMM) are finite set of states, with each state associated with a high dimensional probability distribution [28].. State transitions among them are controlled by a set of probabilities known as transition probabilities. In Figure 4.8, a state transition is shown. In a specific state, an observation or outcome can be found in accordance with the associated probability distribution. The outcome state is not visible externally. Thus, the outsider finds the states as 'hidden'. Generally, Markov models are used to model a large number of real world processes very successfully. Some HMM used processes do not follow the assumption of the dependency of current state

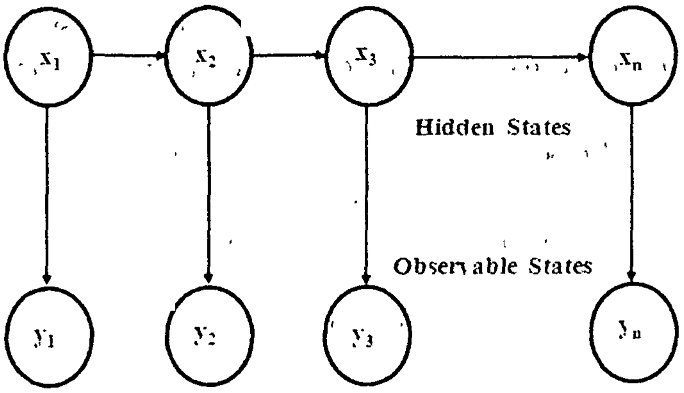


Figure 4.8: Transition states of hidden Markov model (HMM)

from the previous state.

HMM modeling schemes in intrusion detection, consist of observed states, hidden states (intrusion), and HMM profiles. In training HMM uses initial data and re-estimation process creates profile which consists of transition and observed symbol probabilities. Involved steps in HMM modeling are as follows:

(i) Observed states measure is derived analytically or logically from the intrusion indicator test-points. These test-points are spread in the entire system.
(ii) The probability of observation is indicated by the probability matrix of instantaneous observation estimation. This can be estimated by using either parametric model explicitly or non-parametric methods implicitly from data.
(iii) Hidden states are estimated by clustering the homogeneous behaviour components together. These states indicates various different intrusion activities which are identified by administrator.
(iv) Hidden state transition probability matrix are estimated from random data or prior knowledge. The prior knowledge and temporal characteristics of long term relation are an approximate probability of transition of state components from one intrusion state to another.

In[120], Ye uses an HMM to detect intrusions into computer and network systems. In this approach, an HMM is used for representing a temporal normal behaviour profile. The HMM of the normal profile is updated from past data of normal behaviour of the system. The analysis of the observed behaviour is performed and the outcome is used to infer the probability to support the observed behaviour by the HMM of the normal profile. A support of low probability indicates anomalous behaviour which may outcome from intrusive activities.

Yeung et al.[121] describe the use of HMM for anomaly detection on basis of profiling sequences of system call and shell command. Using a forward and backward algorithm, during training, the model estimates the sample probability of an observed sequence. On basis of minimum likelihood, a probability threshold is used among all training sequences to distinguish between normal and anomalous behaviour.

4.5.2 Bayesian Network Based

Bayesian network models probabilistic relationships among interested variables of observation. In data analysis, Bayesian networks using in conjunction with statistical method have several advantages[122]. Bayesian networks can encode interdependent relationships among variables and thus, it can handle missing data situations. Bayesian networks, secondly have capability for representing causal relationships and hence, it can be used for prediction of action consequences. Lastly, Bayesian networks have capability for both causal and interdependent probabilistic relationships and therefore they have possibility in applying to model problems which require combining of prior knowledge and data. A lot of variants of Bayesian networks basic technique are used for network anomaly detection[123]. This technique assumes independence among the various attributes. They are used to find conditional dependencies from various attributes of observation. Several researchers have created anomaly detection models[24,25,26] from Bayesian statistics. Bayesian techniques are also frequently used in classification and suppression of false alarms in network anomaly detection. In[24], Kruegel et al. introduce a multisensor fusion approach. In this approach, different IDS sensor outputs are aggregated and a single alarm is produced. It is assumed that with sufficient confidence, a set of events can not be classified as intrusion by a single anomaly detection technique. Bayesian networks have limitation for actual implementation, though it is used for intrusion detection or in prediction of intruder behaviour in some applications. The accuracy of this method is completely dependent on some assumptions. Typically, these assumptions are based on the behavioural model of the system under view. The deviations from the assumptions may deteriorate its accuracy.

4.5.3 Naïve Bayes

The Naïve Bayes is a probability model on basis of simplified Bayesian Networks[124]. Naïve Bayes model, generally, computes the probability of an end result occurring a number of related evidence variables. The end result given by the probability of an evidence variable is assumed to be independent of the probability of other evidence variables which occurs the similar end result. In the training part, the Naïve Bayes algorithm computes the probabilities of an end result producing a particular attribute and this probability is stored. For each attribute. this is repeated.. In the testing part, the time taken to compute the probability in the worst case for each example of the given class is proportional to the number of attributes. There are few disadvantages of this scheme. As mentioned in[24], the classification capability of Naïve Bayes model is similar to a threshold dependent system that calculate the obtained outputs from the child nodes. Another disadvantage is that the interactions among the child nodes do not occur and only their output have influence in probability of the root node. In incorporating additional information to the root node is difficult because the variables containing the information, directly cannot interact with the child nodes.

Ahirwar et al.[27] proposes a novel method comprising of Naïve Bayes and weighted Radial basis function Network (RBF Network). A radial basis function (RBF) network[125] is a type of artificial neural network. In a two layer neural network, RBFs are embedded, where each hidden unit exhibits a radial activated function. The training of these neural networks are performed to compute posterior probabilities of class membership by use of mixtures of Gaussian basis functions which is separated by hyper-planes.

In[25], Valdes et al. develop an anomaly detection approach that employs the Naive Bayes model to perform intrusion detection on traffic bursts. This model is a part of the EMERALD[126] system and it has ability in potential distributed attack detection. In distributed attacks, each individual attack session is not suspicious alone for generation of an alarm.

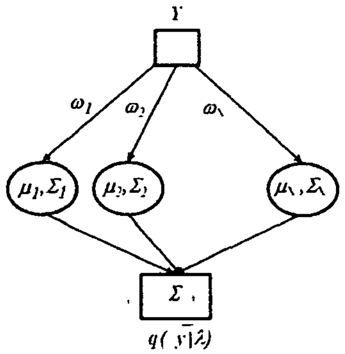


Figure 4.9: Overview of GMM structure

4.5.4 Gaussian Mixture Model

Gaussian Mixture Model (GMM)[30] is a probabilistic learning model. It is a class of density model that includes some component functions, usually called Gaussian The Gaussian mixture density function is a sum of N weighted component densities, as is shown in Figure 4.9 and equation (4.3) indicates the function:

$$q(\bar{y}|\lambda) = \sum_{i=1}^{N} q_i a_i(\bar{y}) \tag{4.3}$$

where $\bar{y}$ is a d-dimensional random vector, $a_i\bar{y}$, $i = 1, 2, \ldots, N$, are the component densities and q_i , $i = 1, 2, \ldots, N$, are the mixture weights. Each component density is a d-dimensional variable Gaussian function of the form,

$$a_i\bar{y} = \frac{1}{(2\pi)^{d/2}\,|\Sigma_i|^{1/2}} exp\left\{-\frac{1}{2}(\bar{y} - \bar{\mu}_i)'\Sigma_i^{-1}(\bar{y} - \bar{\mu}_i)\right\} \tag{4.4}$$

where $\bar{\mu}_i$ is mean and Σ_i is covariance. The mixture weights satisfy the constraint:

$$\sum_{i=1}^{N} q_i = 1. \tag{4.5}$$

The parameters of Gaussian mixture density function are the mean vectors, covariance matrices and mixture weights from all component densities. These parameters are collectively represented by the notation:

$$\lambda = \{q_i, \bar{\mu}_i, \Sigma_i\} \qquad i = 1, 2, \ldots, N. \tag{4.6}$$

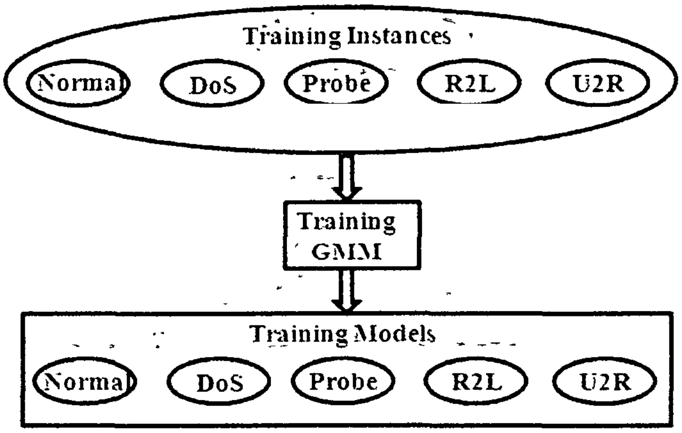


Figure 4.10: Training phase of GMM model

In case of intrusion detection, GMM represents each classification of attacks and it is referred to by model λ. It has different forms depending on the choice of covariance matrices. The model has one covariance matrix per Gaussian component (nodal covariance), one covariance matrix for all Gaussian components in a model (grand covariance), or a single covariance matrix shared by all models (global covariance). The covariance matrix can also be full or diagonal. This allows simplification by using only the diagonal covariance matrix. Bahrololum et al.[31] present an anomaly detection method using GMM method. This process identifies abnormal packets in the network traffic. The method learns statistics of the parameters from traffic packets. Here, all the input sets are framed by itself without comparing to other groups. This approach builds the best possible probability distribution for each group by a set of Gaussian probability distribution functions, i.e., Gaussian Mixtures. The means and variances of the number of mixtures in each model have different effects on performance. In this approach, nodal and diagonal covariance matrices are used for the GMM model. There are two steps in the process: training and detection. The training phase as shown in Figure 4.10, generate reference templates. In the training phase, attacks are trained to the system. Specification of attacks are given to the database for performing the detection process on that specific attack. The features are extracted from the sample data of network traffic to obtain data for modeling of statistical model. In the detection phase, as shown in Figure 4.11, the deviation of input packet is

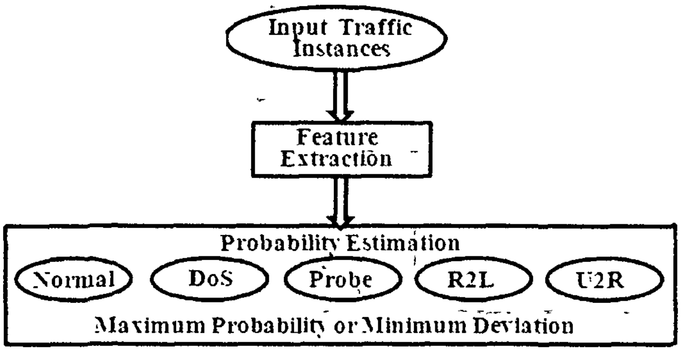


Figure 4.11: Detection phase of GMM model

computed from the reference models of stored data and recognition decision is made to which model the packet suits.

4.5.5 Expectation Maximization Method

Expectation-Maximization (EM)[127] algorithm is a method of obtaining the *maximum likelihood estimation*(MLE)[128] of the parameters from a given dataset where the dataset has incomplete or missing values. Generally, two main applications are available for the EM algorithm. The first application of EM algorithm is when there is missing values in the data because of limitations or problems in the observation process. Secondly, the EM algorithm is applied in optimization the likelihood function. In this situation, optimizing the likelihood function is intractable in analysis but the likelihood function is possible to be simplified by assumption of values for missing (or hidden) parameters. The second EM algorithm application is more commonly used in pattern recognition. In general, EM method is an iterative method. The basic outline of this approach is as follows:

1. Let θ is the current 'best guess' for optimal configuration of a model.

2. Let $\bar{\theta}$ is the next 'best guess' for the optimal configuration of the model.

3 Expectation-Step: Estimate P, the expected value of β with respect to $\bar{\theta}$ for all possible values of hidden parameters The probability of observing each possible set of values of hidden parameter (required to estimate the expectation of β) by using θ.

4. Maximization-Step: Select $\bar{\theta}$ in order to maximize the expected value, P. Then, $\bar{\theta}$ will become the new 'best guess'.

EM cannot estimate at each iteration a target optimal solution. On the other hand, it is possible to estimate a best guess in each iteration to obtain a guaranteed improve. The desirable property of EM method is to converge the solution of the problem to a local optimal solution of the values of θ to improve the value of β, in lieu of converging the solution to a globally optimal one.

Patcha et al.[32] use EM method to cluster the incoming network audit data and estimate the missing values in the data for detection of anomaly. The method begins with an initial guess for the parameters of the GMM (Gaussian Mixture model) for each cluster. The expectation step (E) and the maximization step (M) of EM method are applied iteratively to the clusters, so as to converge to the maximum likelihood fit. In the E-step, the method finds the expected value of the complete data in the provided observed data and estimate parameters. In the M-step, it maximizes the expectation that is estimated in the E-step. These two steps are repeated until occurrence of an increase in the likelihood fit of the data in the current model and the increase value should be less than the threshold of accuracy. The EM algorithm is typically executed multiple times by assuming different initial settings for values of parameters. Thus, a given data point is assigned to the cluster with the largest score of likelihood.

4.6 Soft Computing Methods

Soft Computing constructs computationally intelligent methods individually or in combination of several real problem handling emerging technologies like Fuzzy Logic[33], Probabilistic Reasoning[34], Neural Networks[35], and Genetic Algorithms[36]. These emerging technologies are capable of providing method of reasoning and searching for solving complex and real issues of problems. In comparison to ordinarily used, hard computing, the soft computing is more tolerant of imprecision, uncertainty, and partial truth [37].

4.6.1 Artificial Neural Network (ANN)

The motivation of working on artificial neural network (ANN) is from the inception of the recognition that the working way of computation or estimation of conventional digital computer and the human brain are entirely different [38]. The human brain has ability to arrange its constituent structures called neurons in such a way which can perform certain computations such as recognition of pattern or perception or controlling of motor. This computation is multiple times faster than the similar computation of a fastest digital computer. In neural networks for achieving good performance, massive interconnections of neurons are employed. Thus, neural networks obtain information of the environment through learning process of the system. In the learning process of neural networks, a systematic change occurs in an orderly fashion to obtain the desired objective in the neurons interconnection strengths or synaptic weights However, a large collection neuron interconnections improve the capability of complex computation. An artificial neural network (ANN) is a complex structure of large interconnected artificial neurons which have property of input, output as well as computational features. The artificial neurons are structured in such a way that these are closely linked with learning algorithm which is used for train the network. ANN is frequently employed on data clustering, feature extraction and similarity detection in case of anomaly based network intrusion detection.

Neural Network Intrusion Detector (NNID) [129] describes an off-line anomaly detection method, which applies a modified neural network, back-propagation MLP (Multi-layered Perception). The training of MLP is performed for identifying user's profile and the MLP evaluates the user's commands for possible intrusion at the end of each log session The MLP is a multilayered feed-forward network, It consists of an input layer, one or more hidden layers, and an output layer. The activation patterns are applied to the input layer and the output layer supplies the response of the network. The MLP has objective to assign the input patterns to one of the categories which are represented in accordance to the output of the neural networks such that they can represent the probability of class membership.

4.6.2 Rough Sets

Rough set theory (RST) is an extension of classical set theory. It is used to computing in existence of vagueness or imprecision in data. A rough set is related to working on the boundary regions of a set[130]. Usually, rough sets are used in system of classification where the knowledge of the system is incomplete[131]. In other words, rough set is applied to any classification task to form various classes where each class contains objects which are not distinguishably different. These indistinguishable or indiscernible objects are the basic building blocks (concepts) which are used to build knowledge base about the system or real world in rough set. The concept of these indiscernible objects are referred as rough uncertainty. The rough uncertainty is converted to formulaes which are rough sets.

In pattern recognition, RST is used in plenty. In the work[39], the RST is applied to the network anomaly detection. In this approach, RST is used effectively for anomaly detection with low overhead and high efficiency of detection. Here, RST is employed to extract a set of detection rules with minimal size of normal behaviour model. The normal behaviour model is build from the sequences of system call which are generated during the normal execution of a process. The abnormal operating status of a process is detected by the method and thus, it is reported as possible intrusion. A work[132] using RST on real life intrusion dataset is carried out for intrusion detection.

4.6.3 Fuzzy Logic

Zadeh[33] has been introduced fuzzy logic. Fuzzy logic gives a language with syntax and local semantics for translating qualitative knowledge about a problem that is to be solved. Fuzzy logic is related to fuzzy set theory. A mathematical framework is provided by fuzzy set theory to represent and treat the state of uncertainty, imprecision, and approximate reasoning. In classical set theory, the membership of an element is either 100% or 0% t a set where as in fuzzy set theory, an element can have a partial membership. For example, an element t has a membership function, $\mu P(t)$, that represents the degree to which t belongs to the set P. Other features further define the membership function. For a given fuzzy set P, the core membership function is the (con-

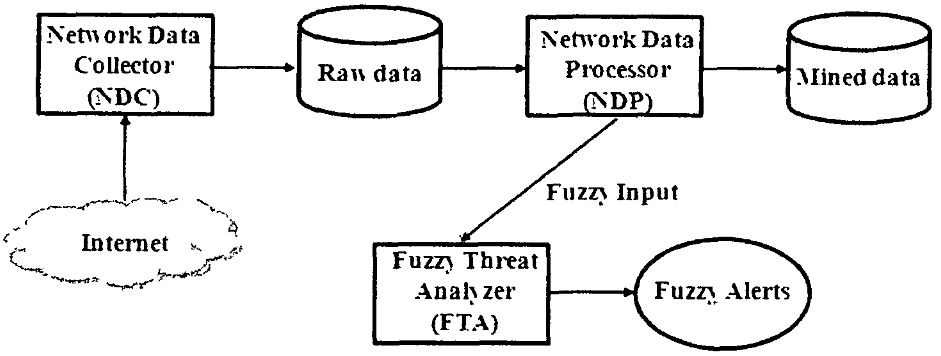

Network Data Collector (NDC)
Raw data
Network Data Processor (NDP)
Mined data
Internet
Fuzzy Input
Fuzzy Threat Analyzer (FTA)
Fuzzy Alerts

Figure 4.12: Architecture of FIRE

ventional) set of all elements $t \in U$ such that $\mu P(t) = 1$. The support of the set is for all $t \in U$ such that $\mu P(t) > 0$ The fuzzy rules are suited to the network security domain. These rules are high level description of patterns of behaviour found in the data. Thus, after identification of the appropriate rules, they are combined using fuzzy reasoning to determine an overall solution.

Dickerson et al. present a fuzzy logic based anomaly intrusion detection method, Fuzzy Intrusion Recognition Engine (FIRE)[133]. FIRE uses fuzzy logic to assess an activity occurred on a network to be malicious or not. Here, simple network traffic metrics are combined with fuzzy rules for determining the likelihood of general or specific network attacks. Thus, these metrics are evaluated as fuzzy sets. The development of fuzzy rules are made for some usually occurred intrusion detection scenarios. Fuzzy network traffic profiles are provided as inputs to its fuzzy rule set. Data mining techniques are applied to process the network input data and develop metrics that are specifically important to anomaly detection. The method uses a fuzzy analysis engine for evaluating the fuzzy inputs and generates security alerts for the administrator. FIRE detects a wide range of common attack types.

The architecture of FIRE is given in Figure 4 12. It consists of the three types of components: network data collector (NDC), network data processor (NDP) and fuzzy threat analyzer (FTA). NDC is used for network data sniffing and recording. It reads raw network packets data and stores them on disk. NDP summarizes and categorizes the raw packet data in tabular format into selected categories. It merges the summaries and tables with past system data. The merged data are stored on disk. NDP also performs comparison of historical mined data with the current data to find values of their difference. These values are 'fuzzified' to generate the fuzzy inputs required by the FTA. The fuzzy inputs from the NDPs represent alert conditions to a degree and thus these are called 'fuzzy alerts'. FTA combines the inputs from one or more NDPs to create composite inputs. Different weights are assigned on the results of the individual NDPs to create different influence on the result. Depending on the fuzzy alerts, fuzzy rules are incorporated to detection intrusion of general or very specific one. The output produced by executing FTA is the fuzzy alerts which are sent to the administrator for response.

4.6.4 Evolutionary Computing

Evolutionary computing or genetic algorithm (GA) is a computational model on basis of principles of evolution and natural selection. In this approach, the problem in a specific domain are converted into a model by use of a chromosome-like data structure. The chromosomes are evolved by using operations like selection, recombination and mutation. Usually, an evolutionary computing process begins with a random selection of population of chromosomes. These chromosomes represent the problem to be solved. Regarding to the attributes of the problem, each chromosome is encoded like bits, characters, or numbers based on their different positions. These positions are also referred to as genes and during the evolution, they are changed randomly within a range. During a stage of evolution, the set of chromosomes are called a population. The 'goodness' of each chromosome are evaluated by using an evaluation function. Crossover and mutation are the two basic operators used during evaluation. Theses two operators are used to simulate the natural reproduction and mutation of species, primarily. The selection process of chromosomes follows the principle of survival of the fittest. In computer network security applications, evolutionary computing is mainly used for finding optimal solutions.

The Applied Research Laboratories of the University of Texas at Austin has developed Network Exploitation Detection Analyst Assistant (NEDAA) [40] in generation of artificial intelligence (AI) rules for ANIDS. The approach uses different machine learning techniques like finite state machine, a decision tree and GA. Here, behaviour of each network connection is converted to represent a rule. This rule is used to judge a connection to be considered either intrusion or not. A collection of such rules are modeled as population of chromosomes. Evolution of the population are performed until the fulfilment of the required criteria. Thus, rules are generated. These rules are used in ANIDS as inside knowledge to judge a network connection behaviour for finding potential intrusions.

4.7 Knowledge Based Methods

Knowledge based methods specifically perform operations like acquisition of knowledge, its representation and application of large bodies of knowledge to a particular problem area. In knowledge based methods [134] for anomaly detection, network activity is checked against pre-defined rules or patterns of normal behaviour. The goal is to employ a representation of normal behaviour (a profile), from which anomalous behaviour is identified as possible attacks. An example knowledge based method is the expert systems approach. An expert system is an knowledge based method. It is a decision making and problem solving method and works on basis of its task knowledge and procedures or logical rules for using the knowledge. Here, the knowledge and the logical rules are obtained from the experts in the domain, generally.

An expert system works in three steps to classify audit data in accordance to a set of rules. At first, different attributes and classes are identified from the audit data. Next, a set of classification rules, procedures or parameters are deduced. Finally, the classification of audit data occur according to the rules. Intrusion Detection Expert System (IDES) [41,42] is the earliest expert system based anomaly detection method. IDES continuously monitors the user behaviour and detects suspicious occurrence. Here, intrusions are flagged in accordance to detection of any deviations from established normal behaviour patterns of individual users Next-Generation Intrusion Detection Expert System (NIDES) [135] is the real time intrusion detection systems. It continuously monitors the user activity and works in a batch mode for periodic analysis of the audit data. A statistical analysis unit maintains a selected set of variables in both IDES and NIDES for creating profile of normal behaviour. This characteristic enables the system to compare the current activity of the user with the desired values of the audited intrusion detection variables that stored in the profile. If the audited activity is significantly deviated from the expected behaviour then it is flagged to be as anomaly. In the stored profile, each variable reflects the behaviour of 'normal conditions'. The computation of each measure/variable is associated to a corresponding random variable. Here, the frequency distribution is produced and updated over time with larger analysis of audit records

4.8 Hybrid Learning Methods

In hybrid learning methods combine of supervised and unsupervised methods of network anomaly detection. Supervised anomaly detection methods perform detection for known type of attacks and it cannot detect unknown attacks. An unsupervised method of anomaly detection has capability for detection of unknown attacks. In a hybrid method, typically, advantages of both supervised and unsupervised methods are combined for improved detection performance. The task of classifier combination is performed variously to overcome deficiencies with one particular classification algorithm. The advantages of multiple classifiers are exploited to overcome their weaknesses or reconciled the outputs from multiple classifiers for handling all situations like known and unknown attack detection. Hybrid method of supervised and unsupervised methods are used in [43,44,45,46,47]. We discuss here some such methods where supervised and unsupervised approaches are incorporated to overcome their weaknesses.

Shon et al.[43] present a hybrid anomaly detection method, Anomaly Detector using Enhanced Support Vector Machines. The entire structure of the anomaly detection framework is parted into of four major steps. The framework is depicted in Figure 4.13. The first step preprocesses and filters traffic using PTF (Passive TCP/IP fingerprinting) [136], and clusters data using Self-Organized Feature Maps (SOFM) [137], and selects packet fields using a Genetic Algorithm (GA). An unsupervised neural network model is used in SOFM for analysis and visualization of high-dimensional data in a two dimensional pattern. The identity of a remote host's operating system is determined by

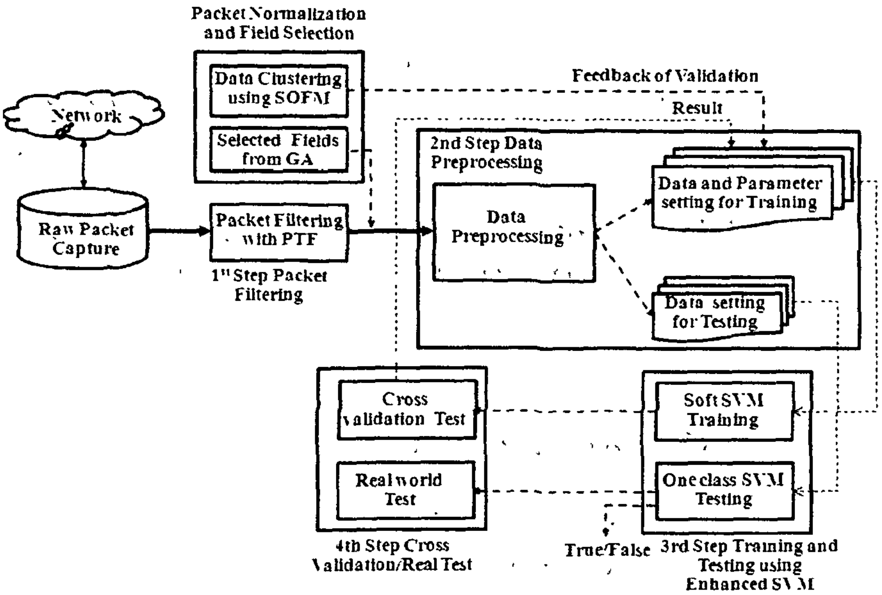

Packet Normalization and Field Selection
Data Clustering using SOFM
Selected Fields from GA
Feedback of Validation
Result
Network
Raw Packet Capture
Packet Filtering with PTF
1st Step Packet Filtering
2nd Step Data Preprocessing
Data Preprocessing
Data and Parameter setting for Training
Data setting for Testing
Cross validation Test
Real world Test
Soft SVM Training
One class SVM Testing
True/False
4th Step Cross Validation/Real Test
3rd Step Training and Testing using Enhanced SVM

Figure 4.13: Enhanced SVM based Anomaly Detection Framework

use of TCP/IP fingerprinting. The PTF is used to drop malicious packets those are not valid for the network. This helps in packet preprocessing and reduces the number of possible attacks on the raw traffic. Before working of overall framework, GA is used for field selection and SOFM are used for packet clustering. Through a simulated evolutionary process, GA selects optimized fields in a packet. The selected fields are used to filter packets in real time. For Support Vector Machine (SVM) packet profiles are created by performing SOFM-based packet clustering. These packet profiles, in turn, create more appropriate training data. The second step preprocess the filtered-packets to obtain high detection performance. Here, the packets which are passed by the previous step are preprocessed with packet relationships on basis of traffic flow and produce inputs for SVM learning. During preprocessing, relationships among the packets are used to associate SVM inputs with temporal characteristics considering IP identification number and using concept of sliding window. In the third step, enhanced SVM machine learning method is included for training and testing. The enhanced SVM model combines two kinds of machine learning methods: supervised soft margin SVM [138] and one-class SVM (unsupervised method) [139]. The enhanced SVM approach has the

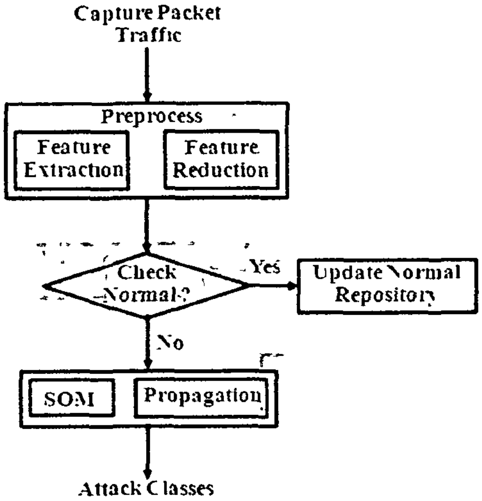


Figure 4.14: Training phase: NN & SOM Hybrid Method

characteristics of high performance of soft margin SVMs, and the novel attack detection capability of the one-class SVMs. Finally, verification of the method is performed using m-fold cross validation test and compare performance with real world NIDSs. Bahrololum et al.[44] introduce a hybrid method using misuse approach for training of normal packets and anomaly detection approach for training of attack packets of network traffic. The approach used for attack training is a combination of supervised and unsupervised Neural Networks (NN). Using the unsupervised NN based method of a Self Organizing Maps (SOMs), attacks are classified into smaller categories considering similar features, and then unsupervised NN based on backpropagation is used for clustering. Self organizing maps[140] use unsupervised learning. Backpropagation[141] is a supervised learning method to teach artificial neural networks. The objective of backpropagation is to train the neural networks for achieving a balance between the correctly responding to input patterns and reasonable response to inputs which are similar to those used in training. The misuse approach identifies known packets fast and unknown attacks are detected by anomaly detection approach. This method of identifying normal and anomalous packets in network traffic work in two phases. The first one is the training phase and it uses a hybrid of SOM and backpropagation NNs. The second one is the detection phase. The frame work of the method is depicted in

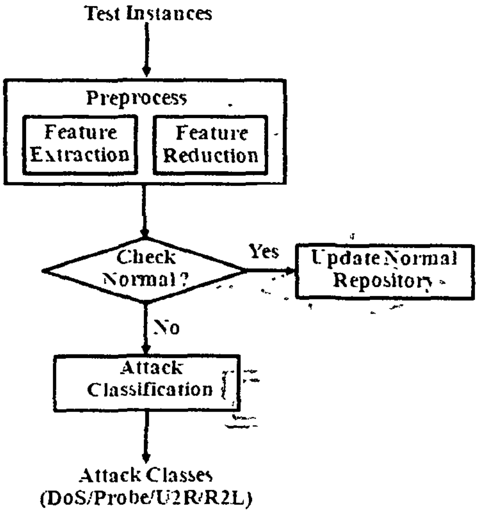


Figure 4.15: Detection phase: NN·&'SOM ·Hybrid Method

Figures 4.14 and 4 15.

As the operations of normal packets are predefined and they exhibit expected behaviour, a knowledge based (misuse-based) IDS can be used for detection of normal activity. However, unexpected activity (presumably an intrusion is unusual) and changes from time-to-time and cannot be detected using knowledge based methods. Therefore, anomaly based intrusion detection supposed to be capable to detect unknown attacks. Using Self Organizing Maps unsupervised NN is used to classify traffic data into smaller categories on basis of similar features and Backpropagation based unsupervised NN is used for clustering traffic data.

The hybrid intrusion detection is an effective model combining the advantages of misuse based intrusion detection and anomaly based intrusion detection method[46]. In[45], Yang et al. present a hybrid intrusion detection method which applies both misuse detection and anomaly detection to a data instance. The architecture of the decision tree based hybrid intrusion detection method is shown in Figure 4.16. An instance is identified as an intrusion if both misuse and anomaly detection approaches detect it as an intrusion. Generalized Stochastic Petri Nets.(GSPN) [142,143] are used to evaluate the performance of the system. The method is'capable of reducing false detection rate and thus it provides a high detection rate. Generalized Stochastic Petri

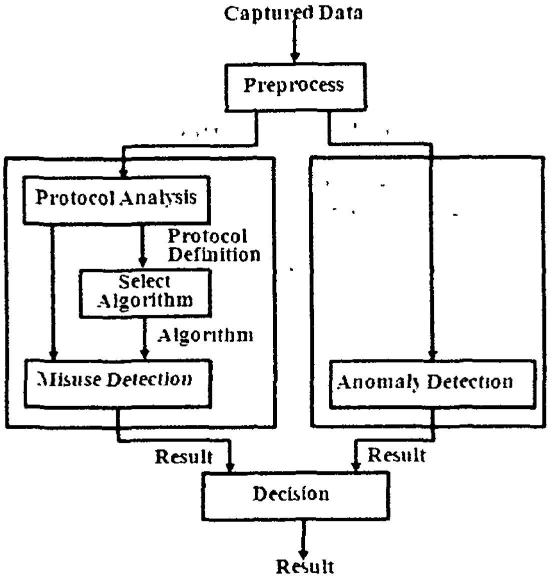


Figure 4.16: Architecture of decision tree based hybrid intrusion detection

Net (GSPN) is a theoretical tool for analysis of system performance. It is an extended version of SPN (Stochastic Petri Net). SPN consists of two kinds of transitions: immediate and timed transition.

4.9 Discussion

In this chapter, we have discussed network anomaly detection methods into six main categories. Usually, obtaining a labeled dataset containing anomalous instances covering all possible types of anomalous behaviour is difficult. Based on availability and use of labeled data, anomaly detection methods are grouped into two. Supervised methods use labeled data for training whereas unsupervised methods perform detection of anomalous without labeled training data. Supervised methods are available for both misuse and anomaly detection. In misuse based methods, predefined patterns are used for intrusion detection whereas anomaly based detection uses profiles of normal or anomalous behaviour in network traffic. Hybrid methods either combine supervised and unsupervised methods or combine anomaly and signature based methods. Supervised anomaly based detection methods are further divided

Table 4 1 List of Existing NIDSs

Name	*Year of Publish*	*Detection Method*	*Detection Time*	*Data Processing*	*Data Gathering*	*Algorithmic Approach*
Snort[144]	1999	M	R	C	C	Rule based
FIRE[143]	2000	A	N	C	C	Fuzzy Logic
ADAM[22]	2001	A	R	C	C	Association Rule
NSOM[145]	2002	A	R	C	C	Neuralnet
MINDS[112]	2003	A	R	C	C	Classification
NTIDS[146]	2003	A	N	C	C	Neuro Fuzzy *Logic*
HIDE[147]	2001	A	R	C	D	Statistical & Neuralnet
ADMIT[148]	2002	A	R	C	C	Clustering
ADWICE[18]	2005	A	R	C	C	Clustering
DNIDS[149]	2007	A	R	C	C	K NN
UNADA[111]	2011	A	N	C	C	Clustering
RELOADED[150]	2005	A	N	C	C	Distance based
LDBSCAN[151]	2008	A	N	C	C	Clustering
Yeung et al Approach[121]	2003	A	N	C	C	HMM method
Bahrololum et al Approach[31]	2008	B	N	C	C	GMM method
Shon et al Approach[43]	2007	B	N	C	C	Enhanced SVM
Patcha et al Approach[32]	2005	A	N	C	C	EM method
NNID[129]	1998	A	N	C	C	ANN method
Li et al Approach[39]	2008	A	N	C	C	RST method
NEDAA[40]	1999	A	N	C	C	GA based
NIDES[132]	1999	A	N	C	C	Expert System
Xiang et al Approach[1]	2004	B	N	C	C	Decision Tree
Depren et al Approach[152]	2005	B	N	C	C	Decision Tree & Neuralnet
Zhang et al Approach[16]	2006	B	N	C	C	Random Forest
Hwang et al Approach[153]	2007	B	N	C	C	Signature based
Hui Lu et al Approach[2]	2009	A	N	C	C	Decision Tree

Note-*M Misuse/A Anomaly/B Both, R Realtime/N Non realtime,C-Centralised/D Distributed into (i) parametric and (ii) non-parametric. Parametric methods estimate parameters from the given data, while non-parametric methods do not assume any knowledge of the underlying distribution of data. Unsupervised anomaly detection methods are presented in three different groups: (i) clustering, (ii) outlier mining and (iii) association rule mining (ARM). These methods attempt to identify network anomalies detection without using labeled training data.

False detection rate tends to be lower in supervised anomaly detection methods but obtaining labeled or purely normal data is a critical issue for supervised methods. The most prevalent advantage of the unsupervised anomaly detection approach is the detection of unknown attacks without any previous knowledge. False detection tends to be higher in unsupervised anomaly detection. On the other hand, hybrid methods provide higher rate of anomaly detection with lower false detection rate as these are combinations of well performing (i) supervised and unsupervised methods or (ii) signature and anomaly based methods.

In Table 4.1, we compare several existing NIDSs based on parameters such as detection method (misuse, anomaly or both), detection time (real time or non real time), nature of processing (centralized or distributed), data gathering mechanism (centralized or distributed) and approach of analysis.

Four different categories of methods are proposed in the thesis for network anomaly detection. The methods are evaluated with available and privately generated intrusion datasets. In *Chapter 5*, we introduce a supervised learning method for network anomaly detection.

CHAPTER 5

Anomaly Detection Using Supervised Approach

Contents

This chapter introduces an effective supervised learning method to achieve best detection performances for known attacks. The method was evaluated with two benchmark and three real life intrusion datasets and has been established to perform very well in comparison to its other competing methods.

5.1 Introduction

A network intrusion can be any exploit of a network that compromises its stability or the security of information stored on computers connected to it. An intrusion detection system gathers relevant data from computers or the network and analyzes them for signs of intrusion. Different approaches such as statistical, probabilistic, machine learning, knowledge based, etc. are used for intrusion detection. Some approaches can perform online, that is, they detect attacks in progress in real time, while unsupervised approaches such as data mining provide after-the-fact clues about the attacks to help reduce the possibilities of future attacks of the same type. In general there are two types of approaches[20] for network intrusion detection : misuse detection and anomaly detection. Misuse detection searches for specific signatures to match, signaling previously known attacks without generating a large number of false alarms. Such methods fail to detect new types of attacks as their signatures are not known. Anomaly detection builds models for normal behaviour and significant deviations from it are flagged as attack.

5.2 Fundamentals of Supervised Classifiers

5.2.1 Problem Formulation

A classifier, which is a function (or model) that assigns a class label to each data item described by a set of attributes, is often needed in classification

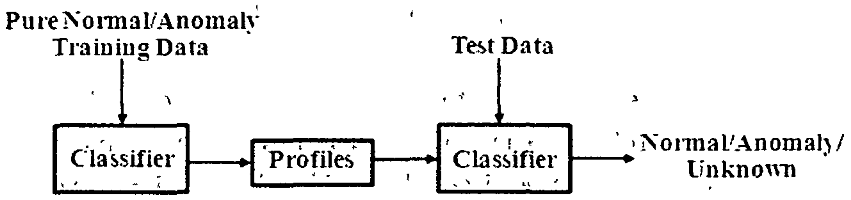


Figure 5.1: A model of supervised classifier

tasks.

For any given test instance $p_i(= x_1, x_2, x_3, \ldots, x_d) \in D_{test}$, i.e., a test dataset for any given training instances $(q_1, q_2, \ldots, q_k) \in Q$, i.e., a training dataset with class variable $c_i \in L$, i.e., each training instance $q_i(= y_1, y_2, y_3, \ldots, y_d, c_i) \in Q$, the job of a supervised anomaly classifier is to identify the appropriate class label c_i for p_i as early as possible with high accuracy.

5.2.2 Supervised Classification

Supervised classification methods require pre-labeled data (or training data), tagged as normal or abnormal. These approaches can be used for classification of unclassified data, where the classifier learns the classification model (viz., profile) and then classifies new exemplars (or test data) as and when required against the learned model. If the new exemplar lies in a region of normality it is classified as normal, otherwise it is flagged as an anomaly or as unknown. Classification algorithms require a good spread of both normal and abnormal labeled data. A generic model of supervised classifier is given in Figure 5.1.

Next, we describe six popular supervised classifiers for network anomaly detection. We also analyze the pros and cons of these classifiers and finally, we compare the performance of our method with these examples.

5.2.2.1 Classification and Regression Tree (CART)

Classification and Regression Trees (CART)[154] is a classification method which uses historical data to construct decision trees Decision trees are then used to classify new data. For building decision trees, CART uses a learning sample - a set of historical data with pre-assigned classes for all observations.

Decision trees are represented by a set of questions which splits the learning sample into smaller and smaller parts. The CART algorithm searches for all possible variables and all possible values in order to find the best split. The process is then repeated for each of the resulting data fragments. The CART methodology consists of three parts: (i) The construction of a maximum tree (ii) The choice of the right tree size and (iii) The classification of new data using constructed tree.

Building the maximum tree implies splitting the learning sample up to the last observations, i.e., when terminal nodes contain observations only of one class. Splitting algorithms are different for classification and regression trees. Classification tree is built in accordance with a splitting rule - the rule that performs the splitting of the learning sample into smaller parts. Each time data is divided into two parts with *maximum homogeneity*. The maximum homogeneity of child nodes is defined by a function called *impurity function* $i(t)$.

Gini splitting rule (or Gini index[155]) is the most commonly used rule. It uses the *impurity function* $i(t)$: $i(t) = \sum_{k \neq l} p(k|t)p(l|t)$, where k and l are indices of the class; $p(k|t)$ and $p(l|t)$ are the conditional probability of class k provided we are in node t.

Unlike a typical classification tree, the regression tree does not deal with classes. In this tree there is a variable matrix Y, and for each observation of this matrix there is a response value. These response values are represented by a vector X, known as the response vector. The regression tree does not use a splitting rule as it does not have classes those are pre-assigned Therefore, the regression tree makes use of the squared residuals minimization algorithm According to this algorithm, the expected sum of variances should be minimum for two resulting nodes. A regression tree is used to analyze data when the outcomes are continuous measurements. A regression tree is created by using recursive partitioning which is the same as in the case of classification tree. Mean-squared error is used to measure the amount of impurity. A terminal node in a regression tree represents the mean of the outcome values contained within the terminal nodes. This value is known as the predicted value for the outcome.

Chebrolu et al. present an IDS in[156] using redundant features in intrusion data. The approach uses two feature selection algorithms involving Classification and Regression Trees (CART) and Bayesian networks (BN) and an ensemble of BN and CART.

5.2.2.2 C4.5

A decision tree is a tree that has three main components: nodes, arcs, and leaves. Each node is labeled with a feature attribute that is most informative among the attributes not yet considered in the path from the root, each arc out of a node is labeled with a feature value for the node's feature and each leaf is labeled with a category or class. Decision tree classifiers are based on the "divide and conquer" strategy to construct an appropriate tree from a given learning set containing a finite and non empty set of labeled instances. The decision tree is constructed during the learning phase and is then used to predict the classes of new instances.

Decision tree learning is machine learning approach for generating classification models. C4.5, a later version of the ID3 algorithm [157], is a decision tree classification algorithm. In ID3, a decision tree is built where each internal node denotes a test on an attribute and each branch represents an outcome of the test. The leaf nodes represent classes or class distributions. The top-most node in a tree is the root node with the highest information gain. After the root node, one of the remaining attributes with the highest information gain is then chosen as the test for the next node. This process continues until all attributes are compared or when all samples are all of the same class or there are no remaining attributes on which the samples may be further partitioned. The attribute with the highest information gain (or greatest entropy reduction) is chosen as the test attribute for the current node. Such an information-theoretic approach minimizes the expected number of tests needed to classify an object and guarantees that a simple tree is found

In[158], Pateriya et al. introduce an intrusion detection system which work based on concept of agent based system. Here, intrusion detection is performed process pattern mining in the system. A copy of the agent is created in the network. This agent resides on the computer which is required to connect to the server. Server agent collects data from client agents to perform process mining and generate alarms on detection of intrusion. The process mining is performed using C4.5 decision tree method.

5.2.2.3 Bayesian Networks

The Bayesian network[159] is a knowledge representation and reasoning algorithm under conditions of uncertainty. A Bayesian network $B = (N, A, \Theta)$ is a Directed Acyclic Graph (DAG) (N, A) where each node $n \in N$ represents a domain variable (e.g., a data set attribute or variable), and each arc $a \in A$ between nodes represents a probabilistic dependency among the variables, quantified using a conditional probability distribution $\theta_i \in \Theta$ for each node n_i. A Bayesian network can be used to compute the conditional probability of one node, given values assigned to the other nodes. Many Bayesian network structure learning algorithms have been developed. These algorithms generally fall into two groups, search and scoring based algorithms and dependency analysis based algorithms. In the area of Bayesian network structure learning, two types of algorithms have been developed. Type 1 deals with a special case where the node ordering is given, which requires $O(N^2)$ conditional independence tests and is correct given that the underlying model is DAG faithful. Type 2 deals with the general case and requires $O(N^4)$ conditional independence tests and is correct given that the underlying model is monotone DAG faithful.

The major advantage of Bayesian networks over many other types of predictive models is that the Bayesian network structure represents the interrelationships among the data set attributes. Human experts can easily understand the network structures and if necessary can easily modify them to obtain better predictive models. By adding decision nodes and utility nodes, BN models can also be extended to decision networks for decision analysis. Other advantages of Bayesian networks include explicit uncertainty characterization, fast and efficient computation, and quick training. They are highly adaptive and easy to build, and provide explicit representation of domain specific knowledge in human reasoning frameworks. Moreover, Bayesian networks offer good generalization with limited training data and easy maintenance when adding new features or new training data.

Wojciech Tylman in[160] proposes an anomaly based intrusion detection system Basset (Bayesian System for Intrusion Detection) using Bayesian Networks (BN). This method enhances the functionality of the open source NIDS, snort[144] to integrate BN as added processing stage. Here, snort is employed for signature based detection and BN is used for anomaly based detection.

5.2.2.4 Naive Bayes

The Naive Bayes classifier computes conditional probabilities to solve a classification problem. It uses Bayes' Theorem[34] with independence assumptions, which assumes that the features are conditionally independent of one another given a class. When a set of classes is observed in the training data, the Naive Bayes classifier then assigns an observed data to a class with the highest probability.

Naive Bayes[161] classifiers are simple Bayesian networks which are composed of DAGs with only one root node (called parent), representing an unobserved node, and several children, corresponding to the observed nodes with strong assumption of independence among child nodes in the context of their parents.

The classification is performed by considenng the parent node to be hidden and computing to which each class an object in the testing set should belong, child nodes represent different attributes specifying the object.

Hence, in the presence of a training set one should only compute conditional probabilities since the structure is unique. Once the network is computed, it is possible to classify any new object using its attribute values using the Baye's rule[34] as shown in Equation (5.1), which is expressed as follows:

$$P(c_i|A) = \frac{P(A|c_i).P(c_i)}{P(A)}, \tag{5.1}$$

where c_i is a possible value and A is the total evidence considering the attribute nodes. The evidence A can be dispatched into pieces of evidence, say a_1, a_2,...,a_n relative to attributes A_1, A_2,...,A_n, respectively. Since Naive Bayes works under the assumption that these attributes are independent, their combined probability is obtained as follows:

$$P(c_i|A) = \frac{P(a_1|c_i).P(a_2|c_i). \;\ldots\; .P(a_n|c_i)}{P(A)}, \tag{5.2}$$

where $P(A)$ is determined by normalization condition.

By applying a Naive Bayes classifier to an intrusion detection task, a set of training network traffic data is used to find the prior probabilities for normal or a known class of attacks. As an unseen network traffic arrives, the classifier then uses Bayes Theorem to decide which class the traffic should belong to.

Panda et al.[162] propose an NIDS framework on basis of Naïve Bayes algorithm. This method of framework creates the patterns of the network services over data sets labeled by the services. With the created patterns, the method detects attacks in the datasets using the classifier based on Naïve Bayes algorithm.

5.2.2.5 CN2

CN2[163] is an induction algorithm. An induction system assists in the task of knowledge acquisition or induction of concept descriptions from examples. The CN2 algorithm is efficient and combines combines the ability to cope with noisy data that the ID3 algorithm[157] has with the if-then rule form and flexible search strategy of the AQ algorithm[164]. The representation for rules output by CN2 is an ordered set of if-then rules. CN2 uses a heuristic function to terminate search during rule construction, based on an estimate of the noise present in the data. This results in rules that may not classify all the training examples correctly, but that perform well on new data.

The CN2 algorithm must make two heuristic decisions during the learning process, and it employs two evaluation functions to aid in these decisions. First it must assess the quality of complexes, determining if a new complex should replace the 'best complex' found so far and also which complexes to discard if the maximum size is exceeded. Computing this involves first finding the set E' of examples which is a complex cover (i.e., which satisfy all of its selectors) and the probability distribution $P = (P_1, \ldots, P_n)$ of examples in E' among classes (where n is the number of classes represented in the training data). CN2 then uses the information-theoretic entropy measure:

$$Entropy = \sum p_i log_2(p_i), \tag{5.3}$$

to evaluate complex quality (the lower the entropy the better the complex). This function thus prefers complexes covering a large number of examples of a single class and few examples of other classes, and hence such complexes score well on the training data when used to predict the majority class covered.

The second evaluation function tests whether a complex is significant. By this we mean a complex that locates a regularity unlikely to have occurred by chance, and thus reflects a genuine correlation between attribute values and classes. To assess significance, CN2 compares the observed distribution among classes of examples satisfying the complex with the expected distribution that would result if the complex selected examples randomly. Some differences in these distributions will result from random variation. The issue is whether the observed differences are too great to be accounted for purely by chance If so, CN2 assumes that the complex reflects a genuine correlation between attributes and classes.

To test significance, the system uses the likelihood ratio statistic [165]. This is given by

$$2\sum_{i=1}^{n} f_i log(f_i/e_i), \tag{5.4}$$

where the distribution $F = (f_1, \ldots, f_n)$ is the observed frequency distribution of examples among classes satisfying a given complex and $E = (e_1, \ldots, e_n)$ is the expected frequency distribution of the same number of examples under the assumption that the complex selects examples randomly. This is considered as the $N = \sum f_i$ covered examples distributed among classes with the same probability as that of examples in the entire training set. This statistic provides an information-theoretic measure of the (noncommutative) distance between the two distributions. This provides a measure that indicates significance. The lower the score, the more likely that the apparent regularity is due to chance.

Thus these two functions - entropy and significance - serve to determine whether complexes found during search are both 'good' (have high accuracy when predicting the majority class covered) and 'reliable' (the high accuracy on training data is not just due to chance). CN2 uses these functions to repeatedly search for the 'best' complex that also passes some minimum threshold of reliability until no more reliable complexes can be found.

Beghdad[166] presents a study of the use of some supervised learning techniques to predict intrusions. The performances of six machine learning algorithms including C4.5, ID3, Classification and Regression Tree (CART), Multinomial Logistic Regression (MLR), Bayesian Networks (BN), and CN2 rule-based algorithm are investigated. KDD Cup 1999 datasets are used to evaluate the algorithms.

5.2.2.6 Support Vector Machines

Support Vector Machine (SVM) is a powerful, state-of-the-art algorithm based on linear and nonlinear regression. Support vector machines (SVM) represent a supervised learning method[167,168]. They perform classification by constructing an N-dimensional hyperplane that optimally separates the data into different categories. In the basic classification, an SVM classifies the data into two categories. Given a training set of instances, labeled pairs (x, y), where y is the label of instance x, an SVM works by maximizing the margin to obtain the best performance in classification.

SVMs are based on the idea of hyper-plane classifiers, or linearly separability. Suppose we have N training data points $(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$, where $x_i \in R^d$ (d-dimensional space or instances) and $y_i \in \{+1, -1\}$. Consider a hyper-plane defined by (w, b), where w is a weight vector and b is a bias. We can classify a new object x with

$$f(x) = sign(w.x + b) = sign\left(\sum_i^n \alpha_i y_i (x_i.x) + b\right). \qquad (5.5)$$

Note that each training vectors x_i occurs a dot product; there is a Lagrangian multiplier[169] α_i for each training point. The Lagrangian multiplier value α_i reflects the importance of each data point. When the maximal margin hyper-plane is found, only points that lie closest to the hyper-plane have $\alpha_i > 0$ and these points are called support vectors. All other points have $\alpha_i = 0$ (see Figure 5.2). This means that only those points that lie closest to the hyper-plane give the representation of the hypothesis/classifier. These most important data points serve as support vectors. Their values can also be used to obtain an independent boundary with regard to the reliability of the hypothesis/classifier. Figure 5.2 shows two classes and their boundaries, i.e., margins. The support vectors are represented by solid objects, while the empty objects are nonsupport vectors. Notice that the margins are only affected by the support vectors, i.e., if we remove or add empty objects, the

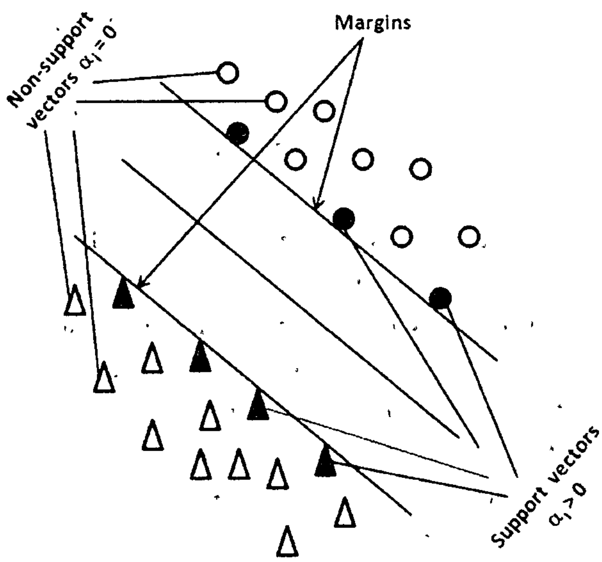

Margins
Non-support vectors $\alpha_i = 0$
Support vectors $\alpha_i > 0$

Figure 5.2: A Value of α_i for support vectors and non-support vectors

margins do not change. Meanwhile any change in the solid objects, either adding or removing objects, could change the margins.

Kim et al.[170] present an NIDS, SVMIDS (Support vector machines based IDS) using SVM. Here, intrusion detection is considered as two-class or multi-class classification problem. The IDS is trained with training data and tested with test data to evaluate novel attacks. The importance of each feature is evaluated to overall improvement of IDS.

5.2.3 More Examples of Supervised Classifiers

Classification is a supervised learning technique that has been applied to anomaly detection. Lee and Stolfo[20] proposed a systematic framework employing data mining techniques for intrusion detection. .This framework consists of classification, association rules and frequent episodes algorithms that can be used to construct detection models. The authors in[171] presented PN-rule, for multi-class classification problem. The key idea used in PNrule is learning a rule-based model in two stages: first find P-rules to predict presence of a class and then find N-rules to predict absence of a class. The scoring mechanism used in PNrule allows one to tune selectively the effect of each N-rule on a given P-rule. ADWICE (Anomaly Detection With fast Incremental Clustering)[18] used extended BIRCH[172] clustering algorithm to implement a fast, scalable and adaptive anomaly detection scheme. They apply clustering as a technique for training of the normality model. Several soft computing paradigms, viz., fuzzy rule-based classifiers, support vector machines, linear genetic programming and an ensemble method to model fast and efficient intrusion detection systems were investigated in[173]. Empirical results clearly show that soft computing approach could play a major role for intrusion detection. Yang et al.[19] present an anomaly detection approach based on clustering and classification for intrusion detection. They perform clustering to group training data points into clusters, from which they select some clusters as normal, and create known-attack profiles according to certain criteria. The training data excluded from the profile are used to build a specific classifier. During the testing stage, they use an influence-based classification algorithm to classify network behaviours. A comparative study of several supervised probabilistic and predictive machine learning techniques for intrusion detection is reported in[174]. Two probabilistic techniques, Naive Bayes and Gaussian, and two predictive techniques, Decision Tree[175] and Random Forests[176], are used. The ability of each technique to detect four attack categories (*DoS, Probe, R2L and U2R*) has been compared. A new supervised intrusion detection method based on TCM-KNN (Transductive Confidence Machine for K-Nearest Neighbours) algorithm is presented in[177]. This algorithm works well even if sufficient attack data are not available for training. A review of the most well known anomaly-based intrusion detection techniques is provided by [48,178]. Available platforms, systems under development and research projects in the area are also presented. The paper also outlines the main challenges to be dealt with for the wide scale deployment of anomaly-based intrusion detectors, with special emphasis on assessment issues. An SVM-based intrusion detection system is found in[179], one which combines a hierarchical clustering algorithm (BIRCH[172]), a simple feature selection procedure, and the SVM (support vector machines) technique. This method was also evaluated using the KDD Cup 1999 datasets.

Table 5.1: List of Existing ANIDSs of Supervised Method

Name	*Year of Publish*	*Detection Method*	*Dataset Used*	*Data Processing*	*Data Gathering*	*Algorithmic Approach*
FIRE[133]	2000	A	K	C	C	Fuzzy Logic
NSOM[145]	2002	A	K	C	C	Neuralnet
MINDS[112]	2003	A	K	C	C	Classification
ADWICE[18]	2005	A	K	C	C	Clustering
TCM-KNN[177]	2007	A	K	C	C	K-Nearest Neighbours
Beghdad et al.[166]	2010	A	K	C	C	CART, C4.5, BN, MLR & CN2
SVM-based[179]	2011	A	K	C	C	SVM & Hierarchical Clustering

Note-*M-Misuse/A-Anomaly/B-Both, K-KDD Cup 1999, C-Centralised/D-Distributed

5.2.4 Discussion

Supervised classifiers are algorithms that reason from externally supplied instances to produce general hypotheses, which then make predictions about future instances. In other words, the goal of supervised learning is to build a concise model of the distribution of class labels in terms of predictor features. The resulting classifier is then used to assign class labels to the testing instances where the values of the predictor features are known, but the value of the class label is unknown. Examples of supervised classification approaches include statistical algorithms, decision tree approaches, proximity based soft computing and many other algorithms in machine learning. Statistical methods are based on assumptions regarding the underlying data distribution. Decision tree learners use a method known as divide and conquer to construct a suitable tree. If there are no conflicting cases, a decision tree correctly classifies all training cases. The major limitation of proximity-based approaches is the selection of an appropriate proximity measure. Soft computing represents combinations of several computational intelligence methodologies which neither are independent nor compete with one another but work in a complementarity. Some of the most commonly used supervised classifiers are CART, C4.5, Bayesian Networks, Naive Bayes, CN2, SVM and soft computing hybrid methods such as fuzzy-neuro. These methods are closely related to types, sizes as well as dimensions of data. Some popular ANIDSs of supervised learning method are given in Table 5.1.

5.2.5 Contributions

An effective supervised anomaly detection method is designed based on similarity measures. In this chapter, emphasize is given on the followings:

- Purely statistical models are useful for trivial cases (e.g., for abrupt change detection).
- Real-time traffic modeling demands for proximity finding over mixed type data with high sensitivity.
- Decision tree learners are generally unstable in nature, due to newer change in the threshold values which affect the result sequentially.
- Clustering based methods are relatively more stable in nature, however, the effectiveness of such methods are highly dependent on what proximity measure used by the method.

The problem of detecting attacks with high detection rate as well as lower false positive rate, poses a special challenge to the research community. Within such a context, intrusion detection using the supervised classifiers have produced remarkable improvements. In this chapter we present a supervised classifier for anomaly-based intrusion detection, which is capable of detecting network attacks. The implementation of supervised classifier uses a combination of supervised as well as unsupervised incremental classification technique for classification of network attacks. We discuss in further details of our method to implement each module and evaluate its performance against a wide range of network attacks.

5.3 Proposed Supervised Method

The proposed supervised method uses a training algorithm to create a set of representative clusters from the available labeled training objects. Unlabeled test objects are then inserted in these representative clusters based on similarity calculations and thus get labels of the clusters in which they are inserted. We first present the basics of the clustering algorithm, and then present the training and prediction algorithms. The clustering algorithm is a modification of the *CatSub* algorithm[180].

5.3.1 Proximity measures

Distance or similarity measures are essential to solve classification, clustering, or retrieval of group of data from a large dataset. From the scientific and mathematical point of view, distance is defined as a quantitative degree of how far apart two objects are. The choice of distance/similarity measures depends on the measurement type or representation of objects. Several number of distance measures viz., *Euclidean, Minkowski* are available[77] for numeric data. Similarly, proximity/similarity measures viz., *Overlap, Eskin* are in exist[78] for categorical data. Similarity measures for mixed type attribute data are specific to domain of data. We introduce two similarity measures for classification of network intrusion data which are of mixed attribute type.

5.3.1.1 Similarity Function Between Two Data Objects

The similarity between two data objects X and Y is the sum of per attribute similarity for all the attributes. It is computed as $sim(X, Y)$,

$$sim(X, Y) = \sum_{j=1}^{d} s(x_j, y_j) \tag{5.6}$$

where $s(x_j, y_j)$ is the similarity for j-th attribute defined as

$$s(x_j, y_j) = \begin{cases} 1, & \text{if } |x_j - y_j| \leq \delta_j \\ 0, & \text{otherwise} \end{cases} \tag{5.7}$$

where δ_j is the similarity threshold for the j-th attribute. For categorical attributes $\delta_j = 0$ and for numeric attributes $\delta_j \geq 0$.

5.3.1.2 Similarity Function Between a Cluster and an Object

The similarity function between a cluster C and an object X becomes $sim(C, X)$

$$sim(C, X) = \sum_{j=1}^{noAttributes} s(v_j, x_{a_j}), \tag{5.8}$$

where $s(v_j,\ x_{a_j})$ is the similarity of the j^{th} attribute of cluster profile and object,

$$s(v_j,\ x_{a_j}) = \begin{cases} 1, & \text{if } |v_j - x_{a_j}| \leq \delta_{a_j} \\ 0, & \text{otherwise .} \end{cases} \tag{5.9}$$

A detailed discussion on selection of appropriate threshold value for δ_j in Equation (5.7) and δ_{a_j} in Equation (5.9) is given in Subsection 5.4.4.

5.3.1.3 Properties of Our Proximity Measure

Similarity or dissimilarity between a cluster and an object is described in terms of the distance between them. In our method, a cluster is represented by a profile; and a profile is similar to an object. A dissimilarity measure, $d(x,y)$, obeys the following four properties (1-4) for any two objects x, y. The similarity measures given in subsections 5.3.1.1 and 5 3.1.2, satisfies all the four properties. The properties[181] are given below.

1. Non-negativity: the distance between any two objects cannot be negative.

2. Identity: the distance between an object and itself must be zero.

3. Symmetry: the distance between object x and y is the same as distance between object y and x, i.e., $d(x,\ y) = d(y,\ x)$.

4. Triangular Inequality: the distance measure should obey the triangle inequality property, i.e., for objects x, y, z, we have $d(x,\ z) \leq d(x,\ y) + d(y,\ z)$.

5.3.2 Background

The dataset to be clustered contains n objects, each described by d attributes $A_1, A_2, \cdots, A_d$ having finite discrete valued domains $D_1, D_2, \cdots, D_d$, respectively. A data object can be represented as $X = \{x_1,\ x_2,\ \cdots,\ x_d\}$. The j-th component of object X is x_j and it takes one of the possible values defined in domain D_j of attribute A_j. Referring to each object by its serial number, the dataset can be represented by the set $N = \{1,\ 2,\ \cdots,\ n\}$. Similarly, the attributes are represented by the set $M = \{1,\ 2,\ \cdots,\ d\}$.

An incremental subspace clustering technique is used here. A cluster is a set of objects which are similar over a subset of attributes only. The minimum size of the subset of attributes required to form a cluster is defined by the threshold $MinAtt$. Let the subset of defining attributes be represented by $Dattributes = \{a_1, a_2, \cdots, a_{noAttributes}\}$ such that $Dattributes \subseteq M$ and $noAttributes$ is the size of $Dattributes$. A cluster is represented by its profile somewhat like an object. All objects in a cluster are similar to the profile. The cluster profile is defined by a set of values, $Values = \{v_1, v_2, \cdots, v_{noAttributes}\}$ considering the attributes in $Dattributes$. That is $v_1 \in D_{a_1}$ is the value for attribute $a_1 \in M$, $v_2 \in D_{a_2}$ is the value for attribute $a_2 \in M$, and so on. Thus, the cluster profile is defined by the similarity function between a cluster C and an object X becomes $sim(C, X)$ as given in Equation (5.8)

$$Profile = \{noAttributes,\ Dattributes,\ Values\}. \tag{5.10}$$

Let $Olist \subseteq N$ be the list of data objects in the cluster. A cluster C is completely defined by its $Profile$ and $Olist$:

$$C = \{Olist,\ Profile\}. \tag{5.11}$$

Our incremental clustering algorithm inserts an object in any one of the clusters existing at a particular moment. So, the similarity between a cluster and a data object needs to be computed. Obviously, the cluster profile is used for computing this similarity. As the similarity is computed over the set of attributes in $Dattributes$ only,

Example 1: Consider a small dataset shown in Table 5.2 with seven objects defined using five attributes A_1, A_2, A_3, A_4 and A_5. The domains for the attributes are $D_1 = \{a1, a2, a3\}$, $D_2 = \{b1, b2\}$; $D_3 = \{c1, c2, c3, c4\}$, $D_4 = \{d1, d2, d3\}$ and $D_5 = \{e1, e2\}$, respectively.

Clusters C_1 and C_2 can be identified in the dataset with $MinAtt = 3$ and $\delta_{a_j} = 0$ for $a_j \in Dattributes$:

$C_1 = \{Olist = \{1, 2, 4\}, noAttributes = 3, Dattributes = \{2, 3, 5\},$
$Values = \{b2, c4, e2\}\}$, and

$C_2 = \{Olist = \{3, 5, 7\}, noAttributes = 4, Dattributes = \{1, 2, 3, 5\},$
$Values = \{a3, b1, c2, e1\}\}$.

Table 5.2: A sample dataset

Serial No.	A_1	A_2	A_3	A_4	A_5
1	a3	b2	c4	d1	e2
2	a2	b2	c4	d3	e2
3	a3	b1	c2	d1	e1
4	a2	b2	c4	d1	e2
5	a3	b1	c2	d3	e1
6	a1	b2	c1	d2	e2
7	a3	b1	c2	d2	e1

5.3.2.1 Our Supervised Algorithms

The supervised classification algorithm starts with an initially empty set of clusters. It reads each object X_i sequentially, inserts it in an existing cluster based upon the similarity between X_i and the clusters or a new cluster is created with X_i if it is not similar enough, as defined by the threshold $MinAtt$, for insertion in an existing cluster. The search for a cluster for inserting the present object is started with the last cluster created and moves towards the first cluster until the search is successful. If successful, the object is inserted in the cluster found and the search is terminated. At the time of inserting the object in the found cluster C, the number of defining attributes of the cluster (*C.noAttributes*) is set according to the computed similarity measure between the cluster and the object and the sets of *C.Dattributes* along with $C.Values$ are updated. If the search is not successful a new cluster is created and the object itself made the representative object of the cluster, i.e., the full set of attributes becomes the *Dattributes* while full set of values of the object becomes corresponding $Values$ of the new cluster profile.

The supervised classification algorithm begins with a fixed number of existing clusters (as defined by their profiles) with no objects present in them. Objects are incrementally inserted in any of the clusters and no new clusters are created.

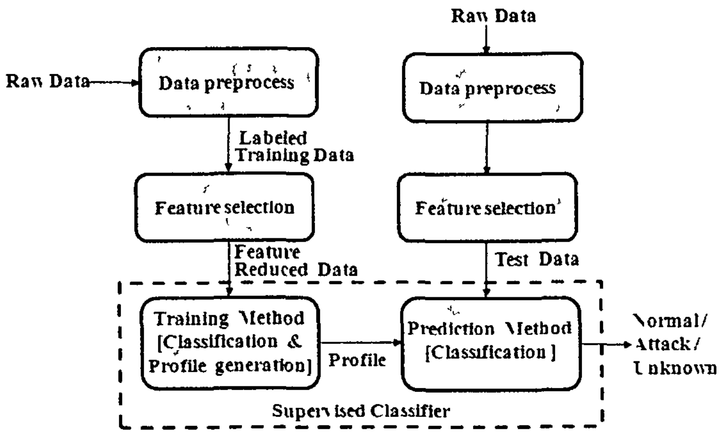


Figure 5.3: Conceptual Framework of Supervised Classifier

5.3.3 Training Method

Given a set of labeled training data, a combination of incremental clustering and supervised classification techniques are applied to create and refine a set of clusters from which profiles are extracted for use in the test object labeling process. The incremental clustering technique decides cluster membership immediately as the objects arrive sequentially without considering subsequent objects that are yet to be seen. Therefore, refinement of the created clusters is performed using a subsequent supervised classification step that allow possible realignment of the data objects in the clusters.

In the beginning of the training algorithm, all objects are marked unprocessed. Similarity thresholds *minAtt* (the minimum number of attributes) and *minSize* (the minimum number of objects in a cluster) are set high and they are gradually decreased in steps In each iteration the remaining unprocessed objects are clustered using a combination of supervised and unsupervised clustering. If the supervised classification process fails to insert an object in any of the preexisting (created in the previous iteration) clusters (SCs), then the unsupervised classification process inserts it in any of the unsupervised clusters (UCs) created in the present iteration or a new unsupervised cluster is created with the object. When the clustering process ends in the present iteration, cluster profiles are extracted from each of the SCs having at least *minSize* objects in it and the objects in such a cluster are marked processed. All SCs are then deleted. The UCs may also include some insignificant clusters whose sizes are less than *minSize*. Such clusters are deleted. Remaining *UC*s are made *SC*s for next iteration after making them empty by deleting their object lists. Then the threshold values *minSize* and *minAtt* are reduced so that the next iteration can create larger clusters instead of fragmented clusters. Reducing the thresholds allows more generalization. The algorithm iterates so long as there are unprocessed objects. To ensure termination of the algorithm *minSize* is reduced to *minSize*/2 so that the ultimate value of *minSize* becomes 1, beyond which no objects remain unprocessed. The threshold *minAtt* is loosened by setting *minAtt* = *minAtt* − α, where α is a small integral constant such as 1 or 2. Reduction of *minAtt* below a certain level (MIN) is not allowed, after which remains constant at MIN. Generalization beyond MIN makes data objects from two different classes indistinguishable. When training terminates, the set of profiles found in the profile file becomes the final cluster profiles for use in the prediction process. The training algorithm is given as Algorithm 1. A training and testing method of supervised classifier is shown in Figure 5.3.

5.3.4 Prediction Method

Once the set of cluster profiles is ready, labeling test objects becomes simple. Supervised classification is performed with the objects. The label of the cluster profile becomes the label of each object inserted in the cluster. The objects not inserted in any of the clusters defined by the profiles need special attention. They may be flagged suspicious for anomalies. Another method for dealing with such objects is to insert such an object in a cluster which is most similar to the object and label accordingly. The algorithm for prediction method is given as Algorithm 4.

5.3.4.1 Effective Profile Searching

Two-dimensional link list (2-DLL) structure (as found in [144]) is used to store the profiles in memory. It enables to update the profile base dynamically. The 2-DLL structure updates row wise with insertion of new parameter in a profile and column wise with the insertion of a new protocol type. It enables to search

Algorithm 1 Training algorithm

```
$file1$=open("TrainProfile","write");
read ($n$, $d$, $minAtt$, $minSize$);
for ($i = 0; i < n; i = i + 1$) do
    read ($X[i]$, $recordLabel[i]$);
    processed[i]=0;
end for
$totalProfile = 0$;
$top = 0$;
while ($minSize > 0$) do                                        ▷ //iterate//
    for ($i = 0; i < n; i = i + 1$) do
        $found = FALSE$;                          ▷ //supervised clustering//
        for ($j = 0; j < totTrain; j = j + 1$) do
            if ($recordLabel[i] == clusterLabel[j]$) then
                if ($sim(SC[j], X[i]) == SC[j].noAttributes$) then
                    $append(i, SC[j].oList)$;
                    $found = TRUE$;
                    break;
                end if
            end if
        end for                                ▷ //end supervised clustering//
        if ($found == TRUE$) then
            continue;
        end if
        for ($j = top; j >= 0; j = j - 1$) do
            if ($recordLabel[i] == clusterLabel[j]$) then
                if ($sim(UC[j], X[i]) >= minAtt$) then
                    $found = TRUE$; $update(UC[j], i, X[i])$;
                    break;
                end if
            end if
        end for
        if ($found == FALSE$) then                  ▷ // create new cluster//
            $top = top + 1$; $UC[top]$ =new $cluster$;
            $append(i, UC[top].oList)$;
            $UC[top].noAttributes = d$;
            for ($j = 0; j < d; ++j$) do
                $UC[top].a[j] = j$;
                $UC[top].v[j] = X[i][j]$;
            end for
            $clusterLabel[top] = recordLabel[i]$;
        end if
    end for                      ▷ //end unsupervised clustering algorithm//
```

2 Training algorithm (continued)

```
    for $(i = 0; i < totTrain; i = i + 1)$ do
        $m$=sizeof($SC[i].oList$)
        if $(m >= minSize)$ then
            $k = SC[i].noAttributes$;
            for $(j = 0; j < k; j = j + 1)$ do
                write($file1$, $SC[i].a[j]$);
            end for
            for $(j = 0; j < k; j = j + 1)$ do
                write($file1$, $SC[i].v[j]$);
            end for
            $p = SC[i].oList$;
            for $(j = 0; j < m; j = j + 1)$ do
                $k = p.getdata(j)$; processed[k]=1; p=p.next
            end for
            delete SC[i];
        end if
        $totTrain = 0$;
        for $(i = 0; i < top; i = i + 1)$ do
            $m$=sizeof($UC[i].oList$)
            if $(m >= minSize)$ then
                $SC[totTrain] = UC.[i]$; delete $UC$
                $UC[i].oList = NULL$; $totTrain = totTrain + 1$;
            else
                delete UC[i];
            end if
        end for
        $minSize = minSize/2$; $MinAtt = minAtt - \alpha$;
        if $(MinAtt < MIN)$ then
            $minAtt = MIN$;
        end if
    end for
end while
function $sim(cluster\ C,\ objectX)$
    $count = 0$; $k = C.noAttributes$;
    for $(j = 0; j < k; j = j + 1)$ do
        $l = C.a[j]$;
        if $(abs(C.v[j] - x[l]) <= \delta[l])$ then
            $count = count + 1$;
        end if
    end for
    return $count$;
end function
```

3 Training algorithm (continued)

```
function $update(cluster\ C,\ int\ r,\ object\ X)$
    $append(r,\ C.Olist)$;
    $count = 1$;
    $m = C.noAttributes$;
    for $(j = 0; j < m; ++j)$ do
        $l = C.a[j]$;
        if $(\text{abs}(s(C.v[j] - x[l]) <= \delta[l])$ then
            $count = count + 1$;
            $C.v[count] = C.v[j]$;
            $C.a[count] = C.a[j]$;
        end if
        $C.noAttributes = count$;
    end for
end function

function $createCluster(cluster\ C,\ int\ r,\ object\ X)$
    $C$ =new $cluster$;
    $append(r,\ C.Olist)$;
    $C.noAttributes = d$;
    for $(j = 0; j < d; ++j)$ do
        $C.a[j] = j$; $C.v[j] = x[j]$;
    end for
end function
```

the profile base protocol wise, viz., TCP, ICMP, UDP, "other" (which profiles are not included the protocols viz., TCP, ICMP or UDP. Once the respective protocol is identified in this 2-DLL structure, further traversal is supported for other parameters matching by following the link list structure.

5.3.4.2 Complexity Analysis

The clustering algorithms require one pass through the set of training examples that currently remain unprocessed. Each training example needs to be compared with existing clusters one after another until it gets inserted in one of the clusters. The similarity computation involves a subset of attributes. Therefore, the clustering process has a complexity $O(ncd)$, where n is the number of training examples, c is the number of clusters and d is the number of attributes. Each of the created clusters needs to be visited to extract its

Algorithm 4 Prediction algorithm

```
$file1$=open(“TrainProfile”,“read”);
$file2$=open(“DataFile”,“read”);
read($file1$, $totTrain$);
for $i = 0; i < totTrain; i = i + 1$ do
    read($file1$, $k$);
    $C[i].noAttributes = k$;
    for $(j = 0; j < k; j = j + 1)$ do
        read($file1$, $C[i].a[j]$);
        for $(j = 0; j < k; j = j + 1)$ do
            read($file1$, $C[i].v[j]$);
            if $count == C[j].noAttributes$ then
                print(“Object” $i$ “ is of category”
                $clusterLabel[j]$);
                $found = 1$; break;
            else
                if $count > max$ then
                    $max = count$; $position = j$;
                end if
            end if
        end for
        if $found == 0$ then
            print(“Object” $i$ “is of category”;
            $clusterLabel[position]$);
        end if
    end for
end for
```

size and profile. Hence, maximum time complexity of one iteration of the training algorithm becomes $O(ncd) + O(c)$. The algorithm performs at most k iterations, where $k = log_2(minSize)$. As $minSize$ is the minimum number of objects for a cluster to be considered significant, it is not large. Overall maximum time complexity of the algorithm is $O(kncd) + O(kc)$.

5.4 Classification of Network Anomalies

In this section we establish the effectiveness of our supervised classification technique on several public benchmark and private real life dataset.

5.4.1 Data Preprocessing

We discretized continuous valued attributes by taking logarithm to the base 2 and then converting to integer. This is done for each attribute value z using the computation: $if\ (z > 2)\ z = int(log_2(z)) + 1$. Before taking logarithm, the attributes which take fractional values in the range [0, 1] are multiplied by 100 so that they take values in the range [0, 100]. Nominal valued attributes are mapped to discrete numeric codes which are nothing but serial numbers beginning with zero for the unique attribute values in the order in which they appear in the dataset. The class label attribute is removed from the dataset and stored separately in a different file. The class labels are used for training and evaluating the detection performance of the algorithm.

5.4.2 Feature Selection

The curse of dimensionality is the challenging issue of intrusion detection [182]. Of these large number of features, it can be monitored for the purpose of intrusion detection that many features are less significant or do not have any importance in attack detection. The elimination of these less significant features enhance the computation accuracy and speed; and thus improve the overall performance of IDS. To overcome the issue of feature selection, methods like *Information gain*[183] and *Mutual Information*[80] usually have been used for intrusion detection. We use *Information gain* method for feature selection in our method as this is widely applied method for the similar purpose with very detection performance. In our method, *Information gain* is computed for each of the discretized attributes. Attributes (such as attribute numbers 7, 9, 15, 18, 20 and 21 in KDD Cup 1999 and NSL-KDD datasets) corresponding to very low information gain are removed from the dataset. It reduces computation time.

5.4.2.1 Information Gain

Let S be a set of training set samples with their corresponding labels. Suppose there are m classes and the training set contains s_i samples of class I and s is the total number of samples in the training set. Expected information needed to classify a given sample is calculated by:

$$I(S_1, S_2, \ldots, S_m) = -\sum_{i=1}^{m} \frac{s_i}{S} log_2(\frac{s_i}{S}). \tag{5.12}$$

A feature F with values $\{f_1, f_2, \ldots, f_v\}$ can divide the training set into v subsets $\{S_1, S_2, \ldots, S_v\}$ where S_j is the subset which has the value f_j for feature F. Furthermore let S_j contain s_{ij} samples of class i. Entropy of the feature F is

$$E(F) = \sum_{j=1}^{v} \frac{S_{1j} + S_{2j} + \ldots + S_{mj}}{S} \times I(S_{1j}, S_{2j}, \ldots, S_{mj}). \tag{5.13}$$

Information gain for F can be calculated as:

$$Gain(F) = I(S_1, S_2, \ldots, S_m) - E(F). \tag{5.14}$$

5.4.3 Parameter Selection

The training algorithm has 3 parameters to be given as inputs: *minAtt*, *minSize* and *MIN*. The parameters *minAtt* takes values in the range [1,*d*], where *d* is the number of attributes. Higher values of *minAtt* corresponds to more specialization leading to a larger number of rules and a lower value of *minAtt* corresponds to more generalization leading to a fewer number of rules. But over generalization can make normal and attack data indistinguishable resulting in more false positives or false negatives. So, a lower limit for *minAtt* is specified using the parameter *MIN*. Now, the possible range of values for *minAtt* becomes [*MIN*,*d*]. To determine the value of *MIN*, information gain[183] is calculated for each attribute in the training dataset. Attributes with very low information gain can be ignored and the number of remaining attributes is set as the value for *MIN*. The parameter *minSize* is the minimum number of data elements received to form a cluster. In some cases, at least one training example for an attack should be considered significant. So, the minimum value for *minSize* is 1. Its value should increase with increase in *minAtt* value. The maximum value for *minSize* can be set to be any multiple of $log_2(n)$, where n is the number of training examples.

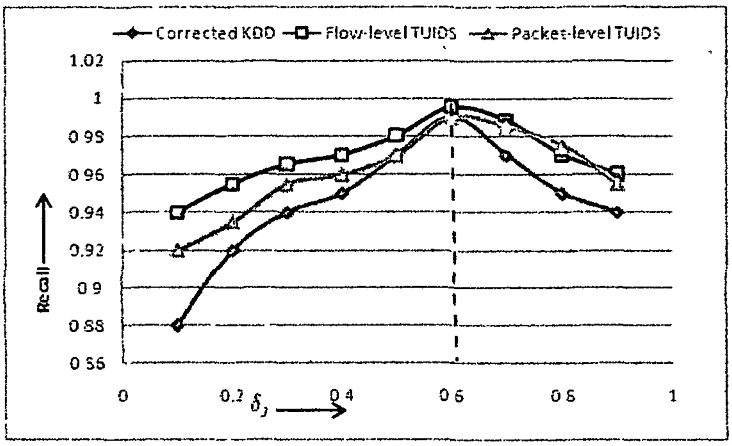


Figure 5.4: δ_j selection

In our experiments, $MIN = 27$. The value for $MinAtt$ is set to 33, which is gradually reduced to $MIN = 27$ in steps of -1. The starting value for $MinSize$ is set to $2log_2(n)$. The similarity thresholds for all attributes are set to zero, $\delta_i = 0$, $i = 1, \cdots, d$.

5.4.4 δ Selection

We have analyzed the effect of selection of δ_j in Equation (5.7) and δ_{a_j} in Equation (5.9) using the benchmark *Corrected* KDD Cup 1999 dataset and real life *Packet level* and *Flow level* TUIDS datasets . The performance of the proposed method in terms of *Recall* depends on the selection of δ_j and δ_{a_j} as seen in Figures 5.4-5.5. It is dependent on the dataset used for evaluation. However, a most probable range of δ_j and δ_{a_j} for these datasets is shown with vertically drawn dashed lines in Figures 5.4 and 5.5. In our experiments, better results are found with $\delta_j = 0.6$ and $\delta_{a_j} = 0.65$.

5.5 Experimental Results

5.5.1 Environment Used

The experiments were performed on a 3 GHz, HP dc 7000 series desktop with 2 GB RAM, 250 GB HDD. C++ programs were used in a LINUX environment. The two benchmark intrusion datasets and three real life intrusion

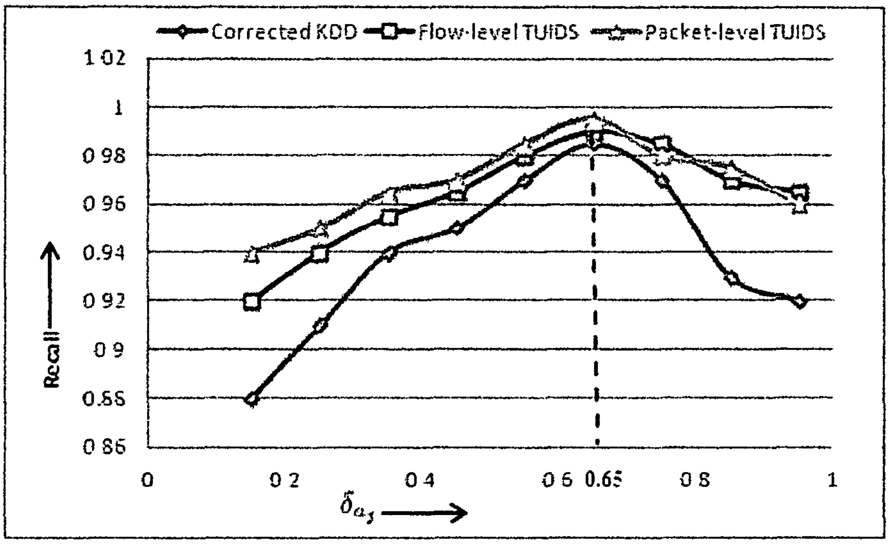


Figure 5.5: δ_{a_j} selection

TUIDS datasets were used to evaluate the performance of the anomaly detection method. We performed three different experiments with separate training to detect anomalies categorized into 2-class (*normal* and *attack*), 5-class (*normal*, *R2L*, *DoS*, *Probe* and *U2R*) and all-attacks behavioural categories. First we provide the results when the same dataset is used for training as well as testing. Then we provide the cross evaluation result performing training and testing of the method with corresponding training and testing datasets. Testbed for supervised classifier is used the framework as shown in Figure 3.5 of *subsection 3.3.2*.

5.5.2 Dataset Used

The algorithm was tested on two benchmark intrusion datasets, viz., KDD Cup 1999[81] and NSL-KDD[82] datasets. The algorithm was also evaluated with three real life TUIDS[92] intrusion datasets. The description of the datasets are given in *Chapter* 3. The Tables 5.3, 5.4 and 5.5 show the attack distributions of the datasets.

Table 5.3: Attack distribution in benchmark intrusion datasets

KDD Cup 1999				NSL-KDD			
Datasets	***Normal***	***Attack***	***Total***	***Datasets***	***Normal***	***Attack***	***Total***
Corrected KDD	60593	250436	311029	$KDDTest^{+}$	9710	12834	22544
10-percent KDD	97278	396743	494021	$KDDTrain^{+}$	67343	58630	125973

Table 5.4: Attack distribution in TUIDS intrusion datasets

Packet Level				Flow Level			
Datasets	***Normal***	***Attack***	***Total***	***Datasets***	***Normal***	***Attack***	***Total***
Training	71785	50142	121927	***Training***	23120	29723	52843
Testing	47895	38370	86265	***Testing***	16770	23955	40725

5.5.3 Results with KDD Cup 1999 Dataset

The experiments are performed for 2-class, 5-class, all-class and cross validation prediction.

5.5.3.1 2-class Prediction Results

The confusion matrices for the 2-class behavioural categories on the *Corrected KDD* and 10 *percent KDD* datasets are shown in Table 5.6 and Table 5.7, respectively. Classification rates of $PCC = 97.57\%$ in Table 5.6 and $PCC = 99.96\%$ in Table 5.7 indicate good performance for our method. Better performance for the second dataset means that data classes in it are well separable compared to the first dataset.

Cross evaluation results when 10-*percent KDD* dataset is used for training and *Corrected KDD* dataset is used for testing is presented in Table 5.8. The classification rate obtained this time is $PCC = 93.40\%$. The performance degrades due to the fact that no examples are present in the training set corresponding to the 15 categories of attacks that are present in the testing set. Most of these attacks are misclassified as *normal* by our algorithm.

Table 5.5: Attack distribution in TUIDS intrusion datasets (continue)

Portscan			
Datasets	***Normal***	***Attack***	***Total***
PortscanTrain	2445	39215	41660
PortscanTest	1300	28615	29915

Table 5.6: 2-class confusion matrix of *Corrected KDD* dataset

	Values	Predicted Class				
		Normal	*Attack*	Sum	Recall	1-Prc*
Actual	*Normal*	54540	6053	60593	0.9001	0.0269
class	*Attack*	1508	248928	250436	0.9940	0.0237
	Sum	56048	254981	311029		
	Re-substitution error rate=0.0243				PCC=97 57%	

Note- *1-Precision

Table 5.7: 2-class confusion matrix of 10-*percent KDD* dataset

	Values	Predicted Class				
		Normal	*Attack*	Sum	Recall	1-Prc*
Actual	*Normal*	97209	69	97278	0.9993	0.0012
class	*Attack*	116	396627	396743	0.9997	0 0002
	Sum	97325	396696	494021		
	Re-substitution error=0.0004				PCC=99.96%	

Note- *1-Precision

Table 5.8: 2-class cross evaluation confusion matrix results of training with 10-*percentKDD* and testing with *Corrected KDD* dataset

	Values	Predicted Class				
		Normal	*Attack*	Sum	Recall	1-Pic*
Actual	*Normal*	60215	378	60593	0.9938	0.2508
class	*Attack*	20155	230281	250436	0:9195	0.0016
	Sum	80370	230659	311029		
	Error=0.0660				PCC=93.40%	

Note- *1-Precision

5.5.3.2 5-class Prediction Results

The confusion matrices for the 5-class behavioural categories on the *Corrected KDD* and 10 *percent KDD* datasets are shown in Table 5.9 and Table 5.10, respectively. The results are found to be almost similar to the 2-class results. The *PCC* for *Corrected* and *10 % Corrected* KDD are 97.57% and 99.96%, respectively. This signifies that the four attack classes have maintained their individual identities to a great extent even after mixing together to form a single attack class.

Cross evaluation results when the 10-*percentKDD* dataset is used for training and the *Corrected KDD* dataset is used for testing are presented in Table 5.11. In this case also most of the unseen attacks are misclassified as normal category and *PCC* reduced to 92.39%.

5.5.3.3 All-attacks Prediction Results

The confusion matrices for all-attacks categories on the *Corrected KDD* and 10-*percent KDD* datasets are shown in Table 5.12 and Table 5.13, respectively. The classification rates still remain nearly similar to the 5-class and 2-class results. The *PCC* for *Corrected* and *10% Corrected* KDD are 97.25% and 99.97%, respectively.

5.5.3.4 Cross Evaluation Prediction Results

Cross evaluation result for the all-attacks case is presented in Table 5.14. In this experiment, the 10 *percent KDD* dataset that includes 22 attack classes is used for training while the *Corrected KDD* dataset with 37 attack classes is used for testing. Since no examples are present in the training set, objects belonging to the 15 categories of attacks cannot be classified correctly and hence we exclude them (18729 objects) from the testing dataset. We see from the results that *DoS* and *Probe* attacks are well detected but *R2L* and *U2R* attacks are detected very poorly. It is apparent from Table 5.13 that there are very few examples in the training set corresponding to attacks belonging to *R2L* and *U2R* categories compared to *DoS*, *Probe* and normal categories.

Table 5.9: 5-class confusion matrix of *Corrected KDD* dataset

	Values	Predicted Class							
		Normal	*R2L*	*DoS*	*Probe*	*U2R*	Sum	Recall	1-Prc*
Actual class	*Normal*	54574	5997	6	15	1	60593	0.9007	0.0273
	R2L	1452	14892	0	1	2	16347	0.9110	0.2873
	DoS	23	0	229828	2	0	229853	0.9999	0.0001
	Probe	42	3	7	4114	0	4166	0.9875	0.0044
	U2R	12	3	0	0	55	70	0.7857	0.0517
	Sum	56103	20895	229841	4132	58	311029		
	Re-substitution error=0.0243					PCC=97.5700%			

Note- *1-Precision

Table 5.10: 5-class confusion matrix of 10-*percent KDD* dataset

	Values	Predicted Class							
		Normal	*R2L*	*DoS*	*Probe*	*U2R*	Sum	Recall	1-Prc*
Actual class	*Normal*	97221	14	1	42	0	97278	0.9994	0.0012
	R2L	88	1036	0	0	2	1126	0.9201	0.0152
	DoS	7	0	391450	1	0	391458	1.0000	0.0000
	Probe	15	0	0	4092	0	4107	0.9963	0 0104
	U2R	6	2	0	0	44	52	0.8462	0.0435
	Sum	97337	1052	391451	4135	46	494021		
		Re-substitution error=0.0004						PCC=99.96%	

Note- *1-Precision

Table 5.11: 5-class cross evaluation confusion matrix results of training with 10-*percentKDD* and testing with *Corrected KDD* dataset

	Values	Predicted Class							
		Normal	*R2L*	*DoS*	*Probe*	*U2R*	Sum	Recall	1-Prc*
Actual class	*Normal*	60211	21	73	284	4	60593	0.9937	0.2614
	R2L	13868	1115	3	1356	5	16347	0.06821	0.0622
	DoS	6360	42	223349	102	0	229853	0.9717	0.0023
	Probe	1036	0	443	2687	0	4166	0.645	0.3933
	U2R	49	11	0	0	10	70	0.1429	0 4737
	Sum	81524	1189	223868	4429	19	311029		
			Error=0.0761					PCC=92.39%	

Note- *1-Precision

Table 5.12: All-attacks confusion matrix of *Corrected KDD* dataset

	Values	Predicted Class			
		Detected	Present	Recall	1-Prc*
Actual class	*normal*	54630	60593	0.9016	0.0257
	snmpgetattack (R2L)	6457	7741	0.8341	0.4792
	named (R2L)	15	17	0.8824	0.3750
	xlock (R2L)	8	9	0.8889	0.1111
	smurf (DoS)	164090	164091	1.0000	0.0000
	ipsweep (Probe)	302	306	0.9869	0.0033
	multihop (R2L)	13	18	0.7222	0.0714
	xsnoop (R2L)	3	4	0.7500	0.0000
	sendmail (R2L)	16	17	0.9412	0.2000
	guess_passwd (R2L)	4358	4367	0.9979	0.0002
	saint (Probe)	692	736	0.9402	0.6121
	buffer_overflow (U2R)	21	22	0.9545	0.0455
	portsweep (Probe)	345	354	0.9746	0.0000
	pod (DoS)	84	87	0.9655	0.0118
	apache2 (DoS)	793	794	0.9987	0.0000
	phf (R2L)	2	2	1.0000	0.0000
	udpstorm (DoS)	2	2	1.0000	0.0000
	warezmaster (R2L)	1561	1602	0.9744	0.0013
	perl (U2R)	2	2	1.0000	0.0000
	satan (Probe)	534	1633	0.3270	0.0582
	xterm (U2R)	10	13	0.7692	0.0000
	mscan (Probe)	1053	1053	1.0000	0.0000
	processtable (DoS)	759	759	1.0000	0.0000
	ps (U2R)	13	16	0.8125	0.0000
	nmap (Probe)	84	84	1.0000	0.0000
	rootkit (U2R)	9	13	0.6923	0.1818
	neptune (DoS)	58000	58001	1.0000	0.0000
	loadmodule (U2R)	2	2	1.0000	0.0000
	imap (R2L)	1	1	1.0000	0.0000
	back (DoS)	1098	1098	1.0000	0.0000
	httptunnel (R2L)	156	158	0.9873	0.0000
	worm (R2L)	0	2	0.0000	1.0000
	mailbomb (DoS)	4999	5000	0.9998	0.0000
	ftp_write (R2L)	3	3	1.0000	0.0000
	teardrop (DoS)	5	12	0.4167	0.1667
	land (DoS)	9	9	1.0000	0.0000
	sqlattack (U2R)	2	2	1.0000	0.0000
	snmpguess (R2L)	2360	2406	0.9809	0.0000
	Sum	302491	311029		
	Re-substitution error=0.0275			PCC=97.25%	

Note- *1-Precision

Table 5.13: All attacks confusion matrix of 10-*percent KDD* dataset

	Values	Predicted Class			
		Detected	Present	Recall	1-Prc*
Actual class	*normal*	97243	97278	0.9996	0.0004
	smurf	280790	280790	1.0000	0.0000
	ipsweep	1243	1247	0.9968	0.0540
	multihop	6	7	0.8571	0.1429
	guess_passwd	53	53	1.0000	0.0000
	buffer_overflow	29	30	0.9667	0.0333
	portsweep	1039	1040	0.9990	0.0000
	pod	261	264	0.9886	0.0000
	phf	4	4	1.0000	0.0000
	warezmaster	18	20	0.9000	0.0000
	perl	3	3	1.0000	0.0000
	satan	1587	1589	0.9987	0.0006
	nmap	180	231	0.7792	0.0164
	rootkit	8	10	0.8000	0.0000
	neptune	107201	107201	1.0000	0.0000
	loadmodule	8	9	0.8889	0.0000
	imap	12	12	1.0000	0.0000
	back	2202	2203	0.9995	0.0000
	ftp_write	6	8	0.7500	0.0000
	teardrop	979	979	1.0000	0.0000
	land	20	21	0.9524	0.0476
	warezclient	991	1020	0.9716	0.0139
	spy	2	2	1.0000	0.0000
	Sum	493885	494021		
	Re-substitution error=0.0003			PCC=99.97%	

Note- *1-Precision

Table 5.14: All-attacks cross evaluation confusion matrix results of training with 10-*percentKDD* and testing with *Corrected KDD* dataset

	Attack Values	Predicted Class			
		Detected	Present	Recall	1-Prc*
Actual class	*normal*	60207	60593	0.9936	0.0790
	smurf	164089	164091	1.0000	0.0002
	ipsweep	298	306	0.9739	0.0418
	multihop	0	18	0.0000	1.0000
	guess_passwd	3	4367	0.0007	0.2500
	buffer_overflow	2	22	0.0909	0.5000
	portsweep	340	354	0.9605	0.2075
	pod	82	87	0.9425	0.1546
	phf	1	2	0.5000	0.0000
	warezmaster	2	1602	0.0012	0.0000
	perl	0	2	0.0000	1.0000
	satan	1280	1633	0.7838	0.1551
	nmap	84	84	1.0000	0.0455
	rootkit	2	13	0.1538	0.9962
	neptune	57971	58001	0.9995	0.0004
	loadmodule	0	2	0.0000	1.0000
	imap	0	1	0.0000	1.0000
	back	1068	1098	0.9727	0.0000
	ftp_write	0	3	0.0000	1.0000
	teardrop	12	12	1.0000	0.7857
	land	6	9	1.6667	0.6667
	warezclient	0	0	0.0000	1.0000
	spy	0	0	0.0000	1.0000
	Sum	285447	292300		
		Error=0.0234		PCC=97.66%	

Note- *1-Precision

Table 5.15: 2-class confusion matrix of $KDDTrain^+$ dataset

	Values	Predicted Class				
		Normal	*Attack*	Sum	Recall	1-Precision
Actual	*Normal*	66793	550	67343	0.9918	0.0021
class	*Attack*	139	58491	58630	0.9976	0.0093
	Sum	66932	59041	125973		
Re-substitution error rate=0.0055				PCC=99.45%		

5.5.3.5 Discussion

The detection performances of our method for *Corrected* and *10 % Corrected* KDD datasets are satisfactory. *PCC* for *10% Corrected* KDD is better than *Corrected* KDD which dataset have many single occurrence attacks. A single occurrence attacks cannot create cluster profile in our method. Thus, *PCC* for *Corrected* KDD is lower than the other one. The cross validation prediction performance for *Corrected* KDD degrades compared to other performances as 15 categories of attacks present in testing dataset *Corrected* KDD which do not exist in the training dataset *10% Corrected* KDD. Most of these attacks are misclassified as normal by our algorithm.

The average execution time of classification for *Corrected KDD* and 10 *percent KDD*, datasets are 1 minute and 1.59 minutes, respectively.

5.5.4 Results with NSL-KDD Datasets

The experiments are performed for 2-class, 5-class and all-class prediction.

5.5.4.1 2-class Prediction Results

The confusion matrices for 2-class behavioural categories on the $KDDTrain^+$ and $KDDTest^+$ datasets are shown in Table 5 15 and Table 5.16, respectively. Classification rates of PCC = 99.45% in Table 5.15 and PCC = 98.34% in Table 5.16 indicate good performance for our technique. The performance for the datasets is better compared to the KDD Cup dataset for 2-class classification.

Table 5.16: 2-class confusion matrix of $KDDTest^+$ dataset

	Values	Predicted Class				
		Normal	*Attack*	Sum	Recall	1-Precision
Actual	*Normal*	9531	179	9710	0.9816	0.0200
class	*Attack*	195	12639	12834	0.9848	0.0140
	Sum	9726	12818	22544		
Re-substitution error rate=0.0166					PCC=98.34%	

Table 5.17: 5-class confusion matrix of $KDDTrain^+$ dataset

	Values	Predicted Class							
		Normal	*R2L*	*DoS*	*Probe*	*U2R*	Sum	Recall	1-Prc*
Actual class	*Normal*	66883	142	114	199	5	67343	0.9932	0.0026
	R2L	47	947	0	0	1	995	0.9518	0.1304
	DoS	23	0	45900	4	0	45927	0.9994	0.0025
	Probe	93	0	0	11563	0	11656	0.9920	0.0173
	U2R	9	0	0	0	43	52	0.8269	0.1224
	Sum	67055	1089	46014	11766	49	125973		
	Re-substitution error=0.0051							PCC=99.49%	

Note- *1-Precision

5.5.4.2 5-class Prediction Results

The confusion matrices for 5-class behavioural categories on the $KDDTrain^+$ and $KDDTest^+$ datasets are shown in Table 5.17 and Table 5.18, respectively. Classification rates of PCC = 99.49% in Table 5.17 and PCC = 98.39% in Table 5.18 indicate good performance for our technioue for both datasets.

5.5.4.3 All-attacks Prediction R

The confusion matrices for all-attacks categories on the $KDDTrain^+$ and $KDDTest^+$ datasets are shown in Table 5.19 and Table 5.20, respectively. Classification rates of PCC = 99.49% in Table 5.19 and PCC = 98.19% in Table 5.20 indicate good performance for our technique. The classification rates still remain better compared to the KDD Cup dataset.

Table 5.18: 5-class confusion matrix of $KDDTest^+$ dataset

	Values	Predicted Class							
		Normal	*R2L*	*DoS*	*Probe*	*U2R*	Sum	Recall	1-Prc*
Actual class	*Normal*	9556	125	5	24	0	9710	0.9841	0.0195
	R2L	157	2727	0	0	3	2887	0.9446	0.0482
	DoS	14	1	7443	0	0	7458	0.9980	0.0009
	Probe	16	5	2	2399	0	2422	0.9905	0.0099
	U2R	3	7	0	0	57	67	0.8507	0.0500
	Sum	9746	2865	7450	2423	60	22544		
		Re-substitution error=0.0161						PCC=98 39%	

Note- *1-Precision

5.5.4.4 Discussion

The detection performances for NSL KDD datasets are better than the results of KDD Cup 1999 dataset. Since both datasets are similar. NSL KDD dataset does not contain any multiple occurrence of instances.

The average execution time of classification for $KDDTrain^+$ and $KDDTest^+$ datasets are 0 41 minute and 0.07 minute, respectively.

5.5.5 Results with TUIDS Intrusion Datasets

The experiments are performed for 2-class and all-class prediction.

5.5.5.1 2-class Prediction Results

The confusion matrices for 2-class behavioural categories on the *Packet Level* and *Flow Level* datasets are shown in Table 5.21 and Table 5.22, respectively. Classification rates of PCC = 99.72% in Table 5.21 and PCC = 99.70% in Table 5.22 indicate good performance for our technique. The performance for the datasets is better compared to the KDD Cup dataset for 2-class classification.

Table 5.19: All-attacks confusion matrix of $KDDTrain^+$ dataset

	Values	Predicted Class			
		Detected	Present	Recall	1-Prc*
Actual class	*normal*	66871	67343	0.9930	0.0014
	smurf	2641	2646	0 9981	0.0403
	ipsweep	3557	3599	0.9883	0.0550
	multihop	4	7	0.5714	0.2000
	guess_passwd	53	53	1.0000	0.0000
	buffer_overflow	27	30	0.9000	0.0000
	portsweep	2929	2931	0.9993	0.0027
	pod	199	201	0.9900	0.0000
	phf	4	4	1.0000	0.0000
	warezmaster	19	20	0 9500	0.0952
	perl	2	3	0.6667	0.0000
	satan	3608	3633	0.9931	0.0014
	nmap	1457	1493	0.9759	0.0714
	rootkit	8	10	0.8000	0.2727
	neptune	41211	41214	0.9999	0.0000
	loadmodule	8	9	0.8889	0.0000
	imap	11	11	1.0000	0.0000
	back	953	956	0.9969	0.0000
	ftp_write	6	8	0.7500	0.0000
	teardrop	892	892	1.0000	0.0000
	land	17	18	0.9444	0.2917
	warezclient	854	890	0.9596	0.0925
	spy	2	2	1.0000	0.0000
	Sum	125333	125973		
	Re-substitution error=0.0051			PCC=99.49%	

Note- *1-Precision

Table 5.20: All attacks confusion matrix of $KDDTest^{+}$ dataset

	Attack Values	*Predicted Class*			
		Detected	Present	Recall	1-Prc*
Actual class	*normal*	9609	9710	0.9895	0.0125
	snmpgetattack	132	178	0.7416	0.3498
	named	15	17	0.8824	0.1176
	xlock	8	9	0.8889	0.1111
	smurf	665	665	1.0000	0.0000
	ipsweep	140	141	0.9929	0.0000
	multihop	17	18	0.9444	0.2273
	xsnoop	3	4	0.7500	0.0000
	sendmail	13	14	0.9286	0.1875
	guess_passwd	1230	1231	0.9992	0.0008
	saint	155	319	0.4859	0.0882
	buffer_overflow	20	20	1.0000	0.0000
	portsweep	149	157	0.9490	0.0688
	pod	39	41	0.9512	0.0000
	apache2	736	737	0.9986	0.0000
	phf	2	2	1.0000	0.0000
	udpstorm	2	2	1.0000	0.0000
	warezmaster	922	944	0.9767	0.0075
	perl	2	2	1.0000	0.0000
	satan	716	735	0.9741	0.1836
	xterm	10	13	0.7692	0.0909
	mscan	997	997	1.0000	0.0000
	processtable	685	685	1.0000	0.0000
	ps	12	15	0.8000	0.0000
	nmap	73	73	1.0000	0.0000
	rootkit	12	13	0.9231	0.2500
	neptune	4656	4657	0.9998	0.0002
	loadmodule	2	2	1.0000	0.0000
	imap	1	1	1.0000	0.0000
	back	359	359	1.0000	0.0000
	httptunnel	132	133	0.9925	0.0000
	worm	2	2	1.0000	0.0000
	mailbomb	292	293	0.9966	0.0034
	ftp_write	3	3	1.0000	0.0000
	teardrop	5	12	0.4167	0.2857
	land	7	7	1.0000	0.0000
	sqlattack	2	2	1.0000	0.0000
	snmpguess	311	331	0.9396	0.0032
	Sum	22136	22544		
	Re-substitution error=0.0181			PCC=98.19%	

Note- *1-Precision

Table 5.21: 2-class confusion matrix for *Packet Level* dataset

	Values	Predicted Class				
		Normal	*Attack*	Sum	Recall	1-Prc*
Actual Class	*Normal*	71268	517	71785	0.9928	0.0017
	Attack	121	50021	50142	0.9976	0.0102
	Sum	71389	50538	121927		
Re-substitution error rate=0 0052					PCC=99.48%	

Note- *1-Precision

Table 5.22: 2-class confusion matrix for *Flow Level* dataset

	Values	Predicted Class				
		Normal	*Attack*	Sum	Recall	1-Prc*
Actual Class	*Normal*	22932	188	23120	0.9919	0.0033
	Attack	75	29648	29723	0.9975	0.0063
	Sum	23007	29836	52843		
Re-substitution error rate=0.0050					PCC=99.50%	

Note- *1-Precision

5.5.5.2 All-attacks Prediction Results

The confusion matrices for all-attacks categories on the *Packet Level* and *Flow Level* datasets are shown in Table 5.23 and Table 5 24, respectively. Classification rates of PCC = 99.42% in Table 5.23 and PCC = 99.01% in Table 5.24 indicate good performance for our technique. The classification rates still remain better compared to the KDD Cup dataset.

5.5.5.3 Discussion

The detection performance results for *Packet Level* and *Flow Level* datasets are very satisfactory. The average execution time of classification for *Packet Level* and *Flow Level* datasets are 0.39 minute and 0.17 minute, respectively.

Table 5.23: All-attacks confusion matrix for *Packet Level* dataset

	Attack Values	Predicted Class			
		Detected	Present	Recall	1-Prc*
	normal	71268	71785	0.9928	0.0014
	smurf	4870	4879	0.9981	0.0403
	1234	2478	2507	0.9883	0.0550
	bonk	48	85	0.5714	0.2000
	fraggle	1086	1086	1.0000	0.0000
	jolt	3484	3485	0.9998	0.0424
Actual class	*nestea*	2993	2995	0.9993	0.0027
	newtear	2965	2995	0.9900	0.0000
	oshare	2520	2520	1.0000	0.0000
	saihyousen	546	575	0.9500	0.0952
	syndrop	6757	6757	1.0000	0.0331
	syn	2549	2566	0.9931	0.0014
	teardrop	1029	1054	0.9759	0.0714
	window	5028	5088	0.9881	0.2727
	winnuke	8599	8599	1.0000	0.0000
	xmas	4906	4951	0.9911	0.0632
	Sum	121126	121927		

Re-substitution error=0.0066 PCC=99.34%

Note- *1-Precision

Table 5.24: All-attacks confusion matrix for *Flow Level* dataset

	Attack Values	Predicted Class			
		Detected	Present	Recall	1-Prc*
Actual class	*normal*	22930	23120	0.9918	0 0018
	smurf	8	13	0.6000	0.0103
	1234	13349	13350	0.9999	0.0150
	bonk	2221	2230	0 9961	0.0340
	fraggle	3124	3128	0.9987	0.0029
	jolt	96	113	0.8486	0 0474
	land	2	2	0.9998	0.0000
	nestea	4	7	0 5217	0.0021
	newtear	9	11	0.7956	0.0062
	saihyousen	16	20	0.7817	0.0059
	syndrop	3	5	0.6515	0.0371
	syn	1660	1699	0.9775	0 0214
	teardrop	6	10	0.5769	0.0314
	window	3345	3402	0 9832	0 3728
	winnuke	2444	2509	0.9740	0.0000
	xmas	3134	3224	0 9720	0.0538
	Sum	52351	52843		
	Re-substitution error=0.0093			PCC=99.07%	

Note- *1-Precision

Table 5 25· 2-class confusion matrix for *PortscanTrain* dataset

	Values	Predicted Class				
		Normal	*Attack*	Sum	Recall	1-Prc*
Actual	*Normal*	2414	31	2445	0.9876	0.2175
Class	*Attack*	671	38544	39215	0.9829	0.0009
	Sum	3085	38575	41660		
	Re-substitution error rate=0 0169				PCC=98.31%	

Note-*1-Precision

Table 5.26: All-attacks confusion matrix for *PortscanTrain* dataset

Values		Predicted Class			
		Detected	Present	Recall	1-Prc*
Actual class	*normal*	2414	2445	0.9876	0.0024
	syn	9612	9750	0.9859	0.0040
	ACK	9875	9945	0.9930	0.0252
	FIN	9551	9780	0.9766	0.0153
	xmas	9505	9740	0.9761	0.0143
	Sum	40957	41660		
Re-substitution error=0.0169			PCC=98.31%		

Note- *1-Precision

5.5.6 Results with TUIDS Portscan Dataset

The experiments are performed for 2-class and all-class prediction.

5.5.6.1 2-class Prediction Results

The confusion matrix for 2-class behavioural categories on the *PortscanTrain* is shown in Table 5.25. Classification rate of PCC = 98.31% in Table 5.25 indicates good performance for our technique.

5.5.6.2 All-attacks Prediction Results

The confusion matrix for all-attack categories in the *PortscanTrain* dataset is shown in Table 5.26. Classification rate of PCC = 98.31% in Table 5.26 indicates good performance for our technique.

5.5.6.3 Discussion

The detection performance results for *Portscan* dataset is very satisfactory. The average execution time of classification for *Portscan* dataset is 0.14 minute.

Table 5.27: Comparison among CART, C4.5, CN2, BN for two-class on *Corrected KDD* dataset

Values	CART		C4.5		CN2		BN		Our Algorithm	
	Recall	**1-Prc***	**Recall**	**1-Prc***	**Recall**	**1-Prc***	**Recall**	**1-Prc***	**Recall**	**1-Prc***
Normal	0.9297	0.0511	0.9297	0.0511	0.8917	0.0640	0.9869	0.1835	0.9001	0.0269
Attack	0.9879	0.0169	0.9879	0.0145	0.9852	0.0259	0.9463	0.0033	**0.9940**	0.0237
PCC	97.65%		97.85%		96.69%		95.42%		97.57%	

Note- *1-Precision

Table 5.28: Comparison among CART, C4.5, CN2, BN for five-class on *Corrected KDD* dataset

Values	CART		C4.5		CN2		BN		Our Algorithm	
	Recall	**1-Prc***	**Recall**	**1-Prc***	**Recall**	**1-Prc***	**Recall**	**1-Prc***	**Recall**	**1-Prc***
Normal	0.9379	0.0526	0.9442	0.0526	0.8708	0.0490	0.7946	0.0624	0.9007	0.0273
R2L	0.7813	0.1919	0.8153	0.1927	0.8451	0.3561	0.8871	0.4554	**0.9110**	0.2873
DoS	0.9984	0.0027	0.9997	0.0011	0.9993	0.0015	0.9811	0.0022	**0.9999**	0.0001
Probe	0.9081	0.2525	0.9482	0.0403	0.9585	0.0230	0.8778	0.3036	**0.9875**	0.0044
U2R	0.5526	0.4545	0.6711	0.0192	0.6754	0.12	0.7281	0.9195	**0.7857**	0.0517
PCC	97.37%		97.83%		96.54%		93.83%		97.57%	

Note- *1-Precision

Table 5.29. Comparison among CART, C4.5, CN2, BN for all Attacks on *Corrected KDD* dataset

Attack values	CART		C4.5		CN2		BN		Our Algorithm	
	Recall	1-Prc*	Recall	1-Prc*	Recall	1-Prc*	Recall	1-Prc*	Recall	1-Prc*
normal	0.9430	0.0540	0.9436	0.0507	0.9822	0.1442	0.7855	0.0215	0.9016	0.0257
snmpgetattack	0.6360	0.3862	0.6384	0.3828	0.0026	0.0000	0.9503	0.6135	0.8341	0.4792
named	0.0000	1.0000	0.2353	0.5000	0.0000	1.0000	0.0000	1.0000	**0.8824**	0.3750
xlock	0.0000	1.0000	0.0000	1.0000	0.0000	1.0000	0.0000	1.0000	0.8889	0.1111
smurf	1.0000	0.0000	1.0000	0.0000	1.0000	0.0011	0.9995	0.0012	**1.0000**	0.0000
ipsweep	0.9641	0.0199	0.9902	0.0000	0.9248	0.0752	0.9739	0.2905	0.9869	0.0033
multihop	0.0000	1.0000	0.0556	0.0000	0.0000	1.0000	0.0000	1.0000	**0.7222**	0.0714
xsnoop	0.0000	1.0000	0.0000	1.0000	0.0000	1.0000	0.0000	1.0000	**0.7500**	0.0000
sendmail	0.0000	1.0000	0.0000	1.0000	0.0000	1.0000	0.0000	1.0000	**0.9412**	0.2000
guess_passwd	0.9968	0.0566	0.9863	0.0242	0.9725	0.0270	0.9679	0.0404	**0.9979**	0.0002
saint	0.1236	0.0000	0.8302	0.2382	0.8003	0.2209	0.7024	0.3277	**0.9402**	0.6121
buffer_overflow	0.0000	1.0000	0.4545	0.4118	0.0000	1.0000	0.0000	1.0000	**0.9545**	0.0455
portsweep	0.8362	0.1111	0.9463	0.1604	0.7260	0.1376	0.9915	0.2252	0.9746	0.0000
pod	0.8391	0.0000	1.0000	0.4082	0.8391	0.0000	0.7701	0.4071	0.9655	0.0118
apache2	0.0000	1.0000	0.9937	0.1841	0.8476	0.0399	0.9786	0.0152	**0.9987**	0.0000
phf	0.0000	1.0000	0.0000	1.0000	0.0000	1.0000	0.0000	1.0000	**1.0000**	0.0000
udpstorm	0.0000	1.0000	0.0000	1.0000	0.0000	1.0000	0.0000	1.0000	**1.0000**	0.0000
warezmaster	0.9863	0.0137	0.9944	0.0293	0.8546	0.1428	0.9850	0.4125	0.9744	0.0013
perl	0.0000	1.0000	0.0000	1.0000	0.0000	1.0000	0.0000	1.0000	**1.0000**	0.0000
satan	0.9724	0.2917	0.8598	0.0628	0.8898	0.1113	0.8677	0.1160	0.3270	0.0582
xterm	0.0000	1.0000	0.2308	0.0000	0.0000	1.0000	0.0000	1.0000	**0.7692**	0.0000
mscan	0.9706	0.0404	0.8927	0.0389	0.8965	0.0268	0.9924	0.0905	**1.0000**	0.0000
processtable	0.9750	0.0250	0.9789	0.0000	0.8762	0.1086	0.9657	0.0639	**1.0000**	0.0000
ps	0.0000	1.0000	0.0000	1.0000	0.0000	1.0000	0.0000	1.0000	**0.8125**	0.0000
nmap	0.0000	1.0000	0.9881	0.3197	1.0000	0.0000	1.0000	0.6147	**1.0000**	0.0000
rootkit	0.0000	1.0000	0.0769	0.0000	0.0000	1.0000	0.0000	1.0000	**0.6023**	0.1818
neptune	0.9990	0.0016	0.9978	0.0011	0.9994	0.0011	0.9923	0.0000	**1.0000**	0.0000
loadmodule	0.0000	1.0000	0.5000	0.0000	0.0000	1.0000	0.0000	1.0000	**1.0000**	0.0000
imap	0.0000	1.0000	0.0000	1.0000	0.0000	1.0000	0.0000	1.0000	**1.0000**	0.0000
back	1.0000	0.4583	0.9545	0.0132	0.7951	0.1164	0.9791	0.0835	**1.0000**	0.0000
httptunnel	0.7025	0.5088	0.8038	0.3553	0.8987	0.1744	0.9494	0.4700	**0.9873**	0.0000
worm	0.0000	1.0000	0.0000	1.0000	0.0000	1.0000	0.0000	1.0000	0.0000	1.0000
mailbomb	0.9516	0.0000	0.9982	0.0062	0.9998	0.0161	0.9992	0.0046	0.9998	0.0000
ftp_write	0.0000	1.0000	0.0000	1.0000	0.0000	1.0000	0.0000	**1.0000**	1.0000	0.0000
teardrop	0.0000	1.0000	0.0000	1.0000	0.0000	1.0000	0.0000	1.0000	**0.4167**	0.1667
land	0.0000	1.0000	0.0000	1.0000	0.0000	1.0000	0.0000	1.0000	**1.0000**	0.0000
sqlattack	0.0000	1.0000	0.0000	1.0000	0.0000	1.0000	0.0000	1.0000	**1.0000**	0.0000
snmpguess	0.9909	0.0004	0.9983	0.0144	0.3603	0.3812	0.3624	0.1718	0.9809	0.0000
PCC		97.25%		97.69%		96.16%		94.75%		97.25%

Note- *1-Precision

Table 5.30: Comparison among CART, C4.5, CN2, BN for two-class on 10 *per cent KDD* dataset

Values	CART		C4.5		CN2		BN		Our Algorithm	
	Recall	**1-Prc***	**Recall**	**1-Prc***	**Recall**	**1-Prc***	**Recall**	**1-Prc***	**Recall**	**1-Prc***
Normal	0.9992	0.0103	0.9978	0.0034	0.9928	0.0037	0.9933	0.0479	**0.9993**	0.0012
Attack	0.9975	0.0002	0.9992	0.0005	0.9991	0.0018	0.9877	0.0017	**0.9997**	0.0002
PCC		99.78%		99.89%		99.78%		98.88%		99.96%

*1-Precision

Table 5.31: Comparison among CART, C4.5, CN2. BN for five-class on 10-*percent KDD* dataset

Values	CART		C4.5		CN2		BN		Our Algorithm	
	Recall	**1-Prc***	**Recall**	**1-Prc***	**Recall**	**1-Prc***	**Recall**	**1-Prc***	**Recall**	**1-Prc***
Normal	0.9987	0.0055	0.9974	0.0040	0.9915	0.0060	0.9825	0.0336	**0.9994**	0.0012
R2L	0.7131	0.0372	0.8188	0.0386	0.8321	0.1235	0.9565	0.4483	**0.9201**	0.0152
DoS	0.9997	0.0002	0.9997	0.0006	0.9993	0.0018	0.9880	0.0017	**1.0000**	0.0000
Probe	0.9511	0.0396	0.9744	0.0133	0.9511	0.0116	0.9645	0.2806	**0.9963**	0.0104
U2R	0.0000	1.0000	0.0577	0.0000	0.4231	0.5319	0.0385	0.9921	**0.8462**	0.0435
PCC		99.83%		99.85%		99.69%		98.65%		99.96%

Note- *1-Precision

Table 5.32: Comparison among CART, C4.5, CN2, BN for all Attacks on 10-*percent KDD* dataset

Values	**CART**		**C4.5**		**CN2**		**BN**		**Our Algorithm**	
	Recall	**1-Prc***	**Recall**	**1-Prc***	**Recall**	**1-Prc***	**Recall**	**1-Prc***	**Recall**	**1-Prc***
normal	0.9967	0.0101	0.9969	0.0035	0.9912	0.0066	0.9769	0.0031	**0.9996**	0 0004
buffer_overflow	0	1.0000	0.0667	0.3333	0	1.0000	0.1	0 6667	**0.9667**	0 0333
loadmodule	0	1.0000	0	1.0000	0	1.0000	0	1.0000	**0.8889**	0.0000
perl	0	1.0000	0	1.0000	0	1.0000	0	1.0000	**1.0000**	0.0000
neptune	1	0.0001	0.9997	0.0001	0.9995	0.0010	0.9993	0.0000	**1.0000**	0.0000
smurf	0.9988	0.0001	1	0.0003	1	0.0015	0.9994	0.0013	**1.0000**	0.0000
guess_passwd	0	1.0000	0.9245	0.0000	0.6981	0.2128	0.9623	0.2917	**1.0000**	0.0000
pod	0.9394	0.2749	0.9811	0.0000	0.6591	0.0000	0.9242	0.3869	**0.9886**	0.0000
teardrop	1	0.0000	1	0.0000	0.9244	0.0812	0.9714	0.2068	**1.0000**	0.0000
portsweep	0.9087	0.0074	0.9663	0.0252	0.9337	0.0433	0 974	0.1130	**0.9990**	0.0000
ipsweep	0.9214	0.0778	0.9374	0.0752	0.935	0.0210	0.9262	0.2366	**0.9968**	0.0540
land	0	1.0000	0.8095	0.1905	0	1.0000	0.5238	0.8103	**0.9524**	0.0476
ftp_write	0	1.0000	0	1.0000	0	1.0000	0	1.0000	**0.7500**	0.0000
back	0.9664	0.0000	0.9664	0.0427	0.9519	0.0981	0.9664	0.2155	**0.9995**	0.0000
imap	0	1.0000	0.25	0.2500	0	1.0000	0.5	0.0000	**1.0000**	0.0000
satan	0.9673	0.04	0.9811	0.0412	0.9025	0.0131	0.8886	0.0201	**0.9987**	0.0006
phf	0	1.0000	0	1.0000	0	1.0000	0	1.0000	**1.0000**	0.0000
nmap	0.3896	0	0.8788	0.2750	0.8052	0.2346	0.4459	0.4663	0.7792	0.0164
multihop	0	1.0000	0	1.0000	0	1.0000	0	1.0000	**0.8571**	0.1429
warezmaster	0	1.0000	0	1.0000	0	1.0000	0	1.0000	**0.0000**	0.0000
warezclient	0.8627	0 1603	0.8745	0.0398	0.8245	0.2036	0.9706	0.4640	**0.9716**	0.0139
spy	0	1.0000	0	1.0000	0	1.0000	0	1.0000	**1.0000**	0.0000
rootkit	0	1.0000	0	1.0000	0	1.0000	0	1.0000	**0.8000**	0.0000
PCC		99.70%		99.83%		99.62%		99 35%		99.97%

Note- *1-Precision

Table 5.33: 5-class attack comparison with SVM-based IDS for *Corrected KDD* Dataset

Values	SVM-based IDS			Accuracy of Our Algorithm
	Correctly Detected	Miss Detected	Accuracy (%)	
Normal	60,166	427	99.29	90.07
DoS	2,28,769	1,084	99.53	**99.99**
Probe	4,064	102	97 55	98.75
U2R	45	183	19.73	78 57
R2L	4,664	11525	28.81	**91.10**
Overall	2,97,708	13,321	95.72	**97.57**

Table 5.34: All class attack comparison with SVM-based IDS for *Corrected KDD* Dataset

Values	Attack present	Attack Detection	
		SVM-based IDS	Our Algorithm
apache2	794	536	793
mailbomb	5000	4459	4999
processtable	759	578	759
mscan	1053	981	1053
saint	736	724	692
httptunnel	158	15	156
ps	16	3	13
sqlattack	2	2	2
xterm	13	5	10
sendmail	17	2	**16**
named	17	3	**15**
snmpgetattack	7741	0	**6457**
snmpguess	2406	1	**2360**
xlock	9	2	**8**
xsnoop	4	0	**3**
worm	2	0	0
Total	18,729	39 04%	**92.56%**

5.5.7 Performance Comparisons

For comparison, the performance rates of some supervised classification algorithms *C4.5*[157], *CART*[154], *Bayesian Network (BN)*[184] and *CN2*[163] rule-based algorithm, as reported in[166] are shown in Tables 5.27 - 5.32 for the *Corrected KDD* and *10 percent KDD* datasets. We see that in terms of *PCC* our algorithm performs better than these algorithms in most of the cases. Only *CART* and *C4.5* performs better than our algorithm in case of the *Corrected KDD* dataset, but in case of *10 percent KDD* dataset, our algorithm performs better than all of these algorithms

The performance rates of SVM-based IDS[179], a combined method of hierarchical clustering and SVM technique, are shown in Tables 5.33 and 5.34 using the KDD Cup 1999 test dataset. The evaluation result using the *Corrected KDD* dataset shows that our algorithm performance rates are distinctly higher in the 5-class attack behaviour and in new attack detection. The accuracy rate of detection for *DoS* attacks is 99.99% as shown in Table 5.33 In the detection of *snmpgetattack* and *snmpguess* attacks, our method detects 6457 and 2360 records respectively, whereas the other method detected 0 and 1 record respectively as shown in Table 5.34.

In this chapter we provide a supervised clustering method and applied it in network anomaly detection. We have developed a subspace based incremental clustering method which forms the basis for the classification method. A training algorithm with a combination of unsupervised incremental clustering and supervised classification algorithm clusters a labeled training dataset into different clusters which are then represented by their profiles. These profiles together with the class labels behave as classification rules. Prediction is done using a supervised classification algorithm that matches testing objects with the cluster profiles for labeling them The effectiveness of the classification method is established on several benchmark and real life TUIDS intrusion datasets. This method has the limitation of attack detection for known attacks.

The unsupervised classification method is applicable for classification of unlabeled data and appropriate for unknown attack detection in network anomaly detection. In the next *Chapter 6*, an unsupervised method is introduced for network anomaly detection.

CHAPTER 6

Anomaly Detection Using Unsupervised Approach

Contents

This chapter is intended to present an unsupervised method to achieve the best detection performances for unknown attacks. The method was evaluated with two benchmark and three real life intrusion datasets and performs very well in comparison to its other competing methods.

6.1 Introduction

With the evolutionary expansion of network-based computer services, the security measure against computer and network intrusions is a crucial issue in a computing environment. The intrusions or attacks to the computer or network system are the activity or attempt to destabilize it by compromising the security in confidentiality, availability or integrity of the system. Rule-based or signature based network intrusion detection methods effectively detect previously known intrusions. In network intrusion detection, usually, threat arises from new or unknown intrusions. Anomaly based intrusion detection approach has the ability to examine new or unknown intrusions. Typically, anomaly based intrusion detection approach builds a model of normal system behaviour from the observed data and distinguishes any significant deviations or exceptions from this model. It implicitly assumes that any deviation from normal behaviour is anomalous. To build or to train a normal behaviour model, the necessary object is the training data which is labeled either as normal or anomalous. These normal behaviour models are used to classify new network connections and gives alert if a connection is classified to be abnormal behaviour. However, in practice, labeled or purely normal data is difficult to obtain and it is error-prone in manual classification for labeling. On other hand, an unsupervised anomaly detection method works without any training data in identifying new or unknown intrusions. These models may be trained on unlabeled or unclassified data and attempt to find intrusions lurked inside the data. Because of this prevalent advantage of unknown intrusion detection without any previous knowledge of intrusions, unsupervised anomaly detection approach is largely popular.

6.2 Fundamentals of Unsupervised Classifier

A supervised anomaly based detection method trains classifier with training data which are labeled as 'normal'. It builds a normal behaviour model. During the testing of the method, any deviations in the testing data from the normal behaviour model is considered as attack. The disadvantage of the supervised method is the difficulty of acquiring guaranteed and labeled training data which is expensive and time consuming in procuring. The unsupervised classifier overcomes the requirement of labeled training data in anomaly detection. They attempt to partition the data and assign natural groups via a similarity measure based on features and properties. These approaches analyze each event or instances to determine how similar (or dissimilar) depending on the choice of similarity measures and dimension weighting. The important features of these classifiers is the capability to learn without knowledge of attack classes or anomalous instances in the data. Thus, they reduce the requirement of labeled training data. Approaches for unsupervised classifier do not assume that the data is labeled. These methods somehow sort the data according to classification. The goal of unsupervised classification to group similar data or objects into subsets. But we are interested in unsupervised anomaly detection to determine the subsets or groups which are most different from the majority of the instances of data. The anomalous groups in a set of instances are not similar and there are many different anomalous groups in one collection of instances or dataset.

6.2.1 Problem Formulation

The unsupervised clustering refers to two concepts that are not necessarily bound to each other. (1) Classification is an operation which basically consists in putting elements in classes. It is often a two fold operation consisting, in (i) learning how to distinguish elements belonging to different classes and, (ii) determining the class belonging to a given element. (2) Unsupervised clustering characterizes a process which is not controlled anyway. A learning is unsupervised if it is made without any training which imposes learning by itself.

With this consideration in view, the problem of unsupervised classification can be stated as follows: given a set of elements $e_1, e_2, \ldots, e_n$, determine clusters $C_1, C_2, \ldots, C_m$ of similar elements without using additional information.

6.2.2 Unsupervised Classification

Unsupervised classification consists of finding classes in a set of elements without using additional information then the data themselves. It causes with minimum input from the operator; no training data is required for training and with identification of natural groupings of the measurement vectors, the feature space subdivision is achieved. It is a categorization process of unclassified data by computer processing solely based on the statistics of attributes without availability of training samples or a prior knowledge of the domain area. Usually, in unsupervised classification, no statistics of the data associated with their class labels are known. Therefore, the goal of unsupervised classifier is to gather the data or objects into groups only on basis of their observable features so as to each group should contain objects which share some important properties. Unsupervised classification method does not know the characteristic of each group or class in advance and it summarizes the characteristics of the group or class after application of unsupervised method. Generally, unsupervised classification methods have two assumptions or rules for the dataset: The number of normal activities are always bigger than the number of anomalous events and there is significant difference between normal and anomalous records. Apart from these assumptions, unsupervised classification approaches have labeling process. In labeling process, group of partitioned data or instances are labeled to be normal or anomalous based on different labeling method viz., indexing. An unsupervised classification approach must use the unlabeled input data to estimate the parameter values for the classification problem at the hand and also to classify the data.

Unsupervised classification or learning approach is similar to the issues of density estimation in statistics[185]. However, unsupervised classification approach includes many other methods that require to summarize and identify key features of the data. A large number of methods used in unsupervised learning are mainly data mining based methods which are used for data preprocessing.

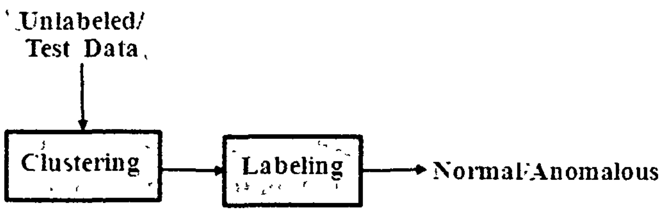


Figure 6.1: A model of unsupervised classifier

6.2.3 Clustering Approaches

Clustering aims to find useful groups of instances of data (clusters) on basis of information found by analysis of the data which describes the relationships among the instances of data of the groups. The goal of the clustering is to attain similarity (or related) among the instances within a group and make differences (or unrelated) among the different groups.

Let $Y = \{Y_1, Y_2, \ldots, Y_n\}$ denote a set of n objects and $Y_i = [y_{i1}, y_{i2}, \ldots, y_{id}]$ be an object represented by d attribute values. Let m be a positive integer. The objective of clustering Y is to find a partition which divides objects in Y into m disjoint clusters.

In cluster analysis, unlabeled instances of data are assigned class label on basis of characteristics of the group developed by a clustering approach. Because this reason, cluster analysis can be referred to as unsupervised classification. Cluster validity analysis is the assessment of output of clustering process. The better and more distinct clustering occurs by creating greater similarity (or homogeneity) within a group and the larger the difference among the groups. Two classes cluster validity measure are often used[186]: Measure of cohesion (or tightness or compactness) which determines the closeness of objects in a cluster, and Measures of cluster separation (or isolation) which determines distinctness or differences of a cluster from other clusters. A generic model of unsupervised classifier is given in Figure 6.1.

Types of clusterings are distinguished to hierarchical versus partitional, exclusive versus overlapping versus fuzzy, and complete versus partial.

Hierarchical versus Partitional:

A hierarchical clustering[53] creates clusters within clusters that it creates a set of nested clusters. In other words, in hierarchical clustering clusters are organized as a tree. Here, each node in the tree is a cluster except for the leaf nodes and each node is the union of its children that is sub clusters. The root of the tree is the cluster which contains all the objects. On the other hand, a partitional clustering is a partition of data objects in a dataset into separated subsets or non-overlapping clusters where each data object belongs to only one cluster or subset exactly.

Exclusive versus Overlapping versus Fuzzy:

In an exclusive clustering[144] each data object is assigned to a single cluster only. There are many situations occur where an object could be reasonably assigned in more than one cluster. In an overlapping or non-exclusive clustering, a single object is justifiably placed in multiple clusters simultaneously. In fuzzy clustering[33], each data object belongs to each of the cluster with a certain membership function value. The membership function value is in between 0 and 1 where 1 represents absolute belonging and 0 represents does not belonging. Thus, clusters are considered as fuzzy sets.

Complete versus Partial:

A complete clustering[22] assigns each data object to a definite cluster. Alternatively, in partial clustering all instances of data objects may not be assigned to clusters that is some of the instances remain unclassified The objective of partial clustering is to assign data objects into well defined clusters. In many datasets, there may have data instances which represent noise or outliers. In those situations, partial clustering excludes them from well defined clusters.

Apart from the above way of categorizing the clustering approaches, another way of classifying the clustering approaches is: partitioning, hierarchical, density based, model based, soft computing and SVM based. Next, we discuss each of these approaches with suitable example system.

6.2.3.1 Partitioning Approach

A partitioning approach separates dataset into separated subsets of data.

k-means based method.

k-means approach is based on k-means algorithm. It is a well-known and widely used simplest clustering algorithm. This algorithm can be used to automatic recognition of groups of similar instances in data. It classifies instances of data to a predefined number of clusters as given by a user for example k clusters. In this approach, the important and first step is to selection of a set of k instances, justifiably. Afterwards, these k instances are considered as centroid (centers of clusters) in the possible clusters. In the next step, the algorithm of the approach read each data instance and compare to each of k instances (i.e., centroid) to assign the current instance to its nearest centroid of cluster. Euclidian distance is the most popular distance measure used to measure distance between centroid and an instance. Although, there are many distance measure methods. After every instance insertion, the centroid of cluster is recalculated and continues it till occurrence of change of centroid.

Portnoy, et al.[70] presents a clustering based unsupervised anomaly detection algorithm in order to detect new intrusions. The training dataset containing unlabeled data is clustered using a modified incremental k-means algorithm. Each cluster is labeled as normal or intrusive based on the number of instances in the cluster. Some percentage of the clusters containing the largest number of instances are labeled as normal and the rest of the clusters are labeled as anomalous. Intrusion in test datasets are detected by using the labeled clusters. The labeling of a test instance is done with the label of its closest cluster.

k-medoid method

k-medoid approach is based on k-medoid algorithm which similar to k-Means clustering algorithm. k-Means aims to minimize the difference between data instance of a cluster and its center (centroid). On the other hand, a medoid is an existing data instance in a cluster which is the central of all data instances in the cluster. k-Means algorithm is high sensitive toward outliers since a data instance with a large value of distance from centroid may distort or biased the data distribution. k-medoid, otherwise, is more robust to outliers or noise present in data as this algorithm perform partitioning on basis of idea of minimization of sum of dissimilarities among data instances in a cluster.

6.2.3.2 Hierarchical Approach

A hierarchical approach builds nested clusters within a cluster.

KNN Approach

KNN (K nearest neighbour) approach of data classification depends on distance or similarity measure defined among data instances in a dataset. Distance (or similarity) among data instances can be estimated in different ways. Euclidean distance is a mostly used distance measure for continuous attribute data. In case of categorical attribute data, 'simple matching coefficient is used and complex distance measure can also be used[78]. In mixed attribute data, usually distance or similarity is computed separately for each attribute and combined them. Here, distance measure always need not to be metric. Typically, the measures are required to be positive, definite and symmetric.

Most of KNN based approaches handle continuous attribute data. A distance measure for mixed attribute data is proposed in [187] for network anomaly detection. In this case, the distance between two instances are computed separately for categorical and continuous attribute and then combined them by defining a link. The number of similar categorical attributes of two instances is the distance between them and a covariance matrix is maintained for continuous attributes to find the dependencies between the continuous values.

Several variants of the basic KNN approach have been proposed to improve the efficiency. In[188], a pruning technique, variant of KNN approach is introduced. In this technique, nearest neighbour distance is computed for each data instance and an anomaly threshold value is set for any data instance to identify the weakest anomaly found so far. The technique discards the close data instances considering as uninteresting.

Ramaswamy et al. proposes a partition based technique in[189], a variant of KNN approach. It initially clusters all the data instances. The lower and upper bounds are then computed on distance of a instance from its k^{th} nearest neighbor for instances in each partition. Then, this information is applied to identify the partitions which possibly cannot contain the top k anomalies. These partitions are pruned. In final phase, from the instances belonging to unpruned partitions, anomalies are computed.

6.2.3.3 Soft computing

Soft computing is a combination of different computing methods. This is an innovative approach and used to build a computationally intelligent system to work in an environment of uncertainty and imprecision by using extraordinary ability of human brain for reasoning and learning[37]. Soft computing, typically, includes several computing paradigms such as neural networks, genetic algorithms, fuzzy sets and probabilistic reasoning. Unsupervised soft computing network anomaly detection works with unlabeled raw network traffic data. However, these methods are based on assumptions of either majority network traffic are of normal or attack/anmalous traffic are rare[190]. Many soft computing approaches have been applied to the unsupervised anomaly detection[84,191,173,192].

Abraham and Jain in[173], uses severel soft computing paradigm such as fuzzy rule-based classifiers, decision trees, support vector machines and linear genetic programming to model an intrusion detection method. Abadeh et al in[193] present a fuzzy genetics-based learning algorithm and describe its usage for intrusion detection in network. Gomez and Dasgupta in [194], utilize genetic algorithm for associating the capability of learning to fuzzy rules. Genetic programming based on Random Subset Selection-Dynamic Subset Selection (RSS-DSS) algorithm[192] for dynamic filtration of dataset is another novel technique exist in network intrusion detection by soft computing.

6.2.3.4 Support Vector Machines Approach

The objective of a support vector machine (SVM)[167,168] is to define a decision hyperplane that separates the different classes with the largest margin from the nearest training examples. The support vectors are the training examples that define the optimal hyperplane, which forms the perpendicular bisector of the support vectors. In essence, the support vectors aim to represent the most informative patterns that allow one to best distinguish between the different classes. SVM based approaches have been widely used in anomaly detection[195,43]. An enhanced SVM approach is proposed in[43] which has characteristics in between the standard supervised SVM and the unsupervised SVM. It behaves as a core component for the hybrid of supervised and unsupervised methods in network anomaly detection. The enhanced SVM approach has inherited the quality of high performance from supervised SVM and the capability of unlabel data detection from unsupervised SVM.

6.2.3.5 More Examples of Clustering Approach

Applying clustering in unsupervised network anomaly intrusion is a wide research area and draw attention from the academic and industrial research community. In[12], a mixture model is presented for detecting the presence of anomalies without training on normal data. This anomaly detection model uses machine learning techniques to compute the probability distributions over data and uses a statistical test to detect anomalies. Old-meadow et al. [196] present a **modified cluster-TV** (time-varying) algorithm based on the **fixed-width clustering**[197] and show improvements in detection accuracy when the clusters are adaptive to changing traffic patterns. Eskin et al. [197] present three algorithms for network anomaly detection: fixed-width clustering algorithm, optimized version of the k-nearest neighbour algorithm (k-NN), and one class support vector machine (SVM) algorithm. In fixed-width clustering, clusters are created based on defined distance in between data objects to isolate smaller cluster for identifying as anomalous. In[66], Kingsly et al. presents a new density and grid based clustering algorithm, **fpMAFIA** based on the subspace clustering algorithm pMAFIA[198]. Grid-based methods divide the object space into a finite number of cells that form a grid structure. All of the clustering operations are performed on the grid structure.

6.2.4 Outlier Based Approach

Outliers are specific patterns in data that do not obey the rules followed by the majority of instances in data. Outlier detection and clustering are closely related. From clustering point of view, outliers are objects that do not located in the clusters of a data set, and with reference to anomaly detection, they may be considered as attacks or anomalies. The concept of outliers are studied from a different view point in[103]. Details are explained extensively as a different category of anomaly detection in *Chapter* 7.

Zhang et al.[193] present an unsupervised outlier based network anomaly detection method using random forests[200] algorithm. This algorithm is result of an ensemble of un-pruned classification or regression trees and suitable on large datasets with many number of features for accurate and efficient results. Random forests algorithm usually, generates many classification trees. Each of the tree is constructed by using a tree classification algorithm from the original data. After construction of the forest, new object is put down each of the tree for classification. The decision of each tree is provided as a vote indicating the class for the object. The most votes procured class for the object is selected by forest. Thus, random forests algorithm detect outliers in network traffic data without attack free training data. In the framework of the outlier detection method, random forests algorithm build network service patterns over traffic data. With reference to the built patterns, outliers are determined for detection of intrusions.

6.2.5 Discussion

Unsupervised classification or learning methods analyze each event to determine how similar (or dissimilar) depends on the choice of similarity measures or dimension weighting. The important feature of unsupervised classification method is its possibility of learning without knowledge of attack class labels. Thus, it reduces the requirement of training data. The fundamental task of unsupervised learning is automatic developing of classification labels. Unsupervised learning algorithms always find out similarity among pieces of data with aim to determine their importance in formation of a group or cluster. In labeling phase of unsupervised classification, these groups are identified as normal or anomalous. Examples of unsupervised classification approaches include clustering, outlier detection, k-Means, k-Medoids, KNN, soft computing, support vector machines and many other algorithms in machine learning. Usually, unsupervised learning algorithms for anomaly detection have two assumptions about the data. The first one assumes normal instances in a dataset are significantly more than intrusive ones. The second one assumes intrusive instances in a dataset are qualitatively different from normal ones. The major advantage of unsupervised learning algorithms of anomaly detection is its capability to process unlabel data and detection of unknown intrusions. The limitation of unsupervised learning algorithms of anomaly detection is the holding of assumptions over the data to be true.

6.2.6 Contributions

In this work we develop a clustering based algorithm k-point for unsupervised anomaly-based detection of intrusions. We evaluate the approach with our generated intrusion TUIDS dataset and well-known intrusion dataset KDD Cup 1999[81] and NSL-KDD dataset[82]. The NSL-KDD dataset is a filtered dataset from KDD Cup 1999 dataset. We compare results of our approach with similar existing approaches.

6.3 Proposed Clustering Method

The proposed clustering method uses the k-point algorithm to create a set of representative clusters from the available unlabeled objects of data. Initially, the method consider k objects randomly from the dataset. The dataset objects are then gathered to these selected points (or objects) based on various attribute similarities. The clusters are formed using two similarity measures: (i) similarity between two objects, and (ii) similarity between a cluster and an object. The method will produce various clusters. The component of a conceptual framework of unsupervised classifier and the detail explanation of the k-point algorithm are presented in the following subsections.

6.3.1 Conceptual Framework

A conceptual framework of unsupervised classifier is given in Figure 6.2. The framework of the classifier consists of the following four phases.

(i) ***Parameterizations:*** The attributes of the input data are analyzed and number of attributes are reduced on basis of frequency occurrence or other

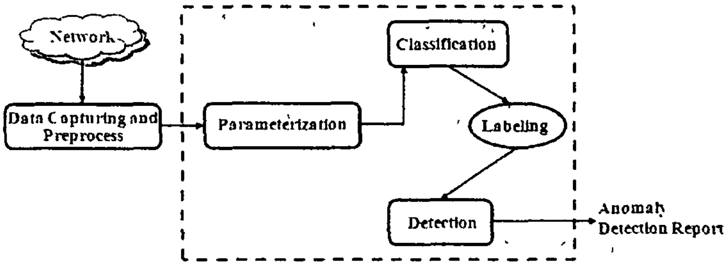


Figure 6.2: A framework of unsupervised classifier

attribute reduction method.

(ii) ***Classification algorithm:*** Classification or clustering algorithms are applied to group the similar instances using similarity measure or other distance measure methods.

(iii) ***Labeling:*** Classification generated groups are labeled either as normal or anomalous using different labeling scheme viz., indexing.

(iv) ***Detection:*** Labeled groups or classes are detected and results of detection rate or other performance measuring metrics are estimated.

6.3.2 k-point Algorithm Fundamental

The dataset to be clustered contains n objects, each described by d attributes $A_1, A_2, \dots, A_d$ having finite valued domains $D_1, D_2, \dots, D_d$ respectively. A data object can be represented as $X = \{x_1, x_2, \dots, x_d\}$. The j-th component of object X is x_j and it takes one of the possible values defined in domain D_j of attribute A_j. Referring to each object by its serial number, the dataset can be represented by the set $N = \{1, 2, \dots, n\}$. Similarly, the attributes are represented by the set $M = \{1, 2, \dots, d\}$.

6.3.2.1 Similarity function between two data objects

Similarity between two data objects X and Y is the sum of per attribute similarity for all the attributes. It is computed as $simO(X, Y)$,

$$simO(X, Y) = \sum_{j=1}^{d} s(x_j, y_j) \tag{6.1}$$

where $s(x_j, y_j)$ is the similarity for j-th attribute defined as

$$s(x_j, y_j) = \begin{cases} 1 & \text{if } x_j \neq y_j \\ 0 & \text{otherwise} \end{cases} \tag{6.2}$$

6.3.2.2 Similarity between a cluster and an object

A cluster is a set of objects which are similar over a subset of attributes only. The minimum size of the subset of attributes required to form a cluster is defined by the threshold *MinmAtt*. Let, the subset of defining attributes be represented by $D_{attrib} = \{a_1, a_2, \ldots, a_{N_{attrib}}\}$ such that $D_{attrib} \subseteq M$ and N_{attrib} is the size of D_{attrib}. A cluster will be represented by its profile that looks like an object. All the objects of the cluster is similar with respect to the profile. The cluster profile is defined by a set of values, $V_{attrib} = \{v_1, v_2, \ldots, v_{N_{attrib}}\}$ taken over the corresponding attributes in D_{attrib}, that is $v_1 \in D_{a_1}$ is the value for attribute $a_1 \in M$, $v_2 \in D_{a_2}$ is the value for attribute $a_2 \in M$ and so on. Thus, the cluster profile is defined by:

$$Clusprofile = \{N_{attrib}, D_{attrib}, V_{attrib}\} \tag{6.3}$$

Let, *Oblist* $\in N$ is the list of data objects in the cluster. A cluster C is completely defined by its *Clusprofile* and *Oblist*:

$$C = \{Oblist, Clusprofile\} \tag{6.4}$$

The k-point clustering algorithm inserts an object in any one of the set of clusters existing at the particular moment. So the similarity between a cluster and a data object needs to be computed. Obviously, the cluster profile is used for computing this similarity. A cluster profile is defined by *equation* (6.3). As the similarity needs to be computed over the set of N_{attrib} attributes in D_{attrib}, positions of V_{attrib} values only of a cluster and an object, the cluster profile is needed to find in the object. The similarity function between a cluster C and an object Y becomes *simC(C, Y)*.

$$simC(C, Y) = \sum_{j=1}^{N_{attrib}} s(x_j, y_j) \tag{6.5}$$

Table 6.1: A sample dataset

Serialno.	A_1	A_2	A_3	A_4	A_5	A_6
1	a_3	b_2	c_4	d_1	e_2	f_1
2	a_2	b_2	c_4	d_3	e_2	f_2
3	a_3	b_1	c_2	d_1	e_1	f_2
4	a_2	b_2	c_4	d_1	e_2	f_1
5	a_3	b_1	c_2	d_3	e_1	f_3
6	a_1	b_2	c_1	d_2	e_2	f_1
7	a_3	b_1	c_2	d_2	e_1	f_3
8	a_2	b_2	c_4	d_3	e_2	f_2
9	a_3	b_1	c_2	d_1	e_1	f_4
10	a_3	b_1	c_2	d_1	e_1	f_5

where N_{attrib} is the attribute numbers in cluster profile and $s(x_j, y_j)$ is the similarity between j-th value x_j of profile of cluster C and j-th attribute value y_j of object Y is defined buy *equation 6.2.*

Example: Consider a sample dataset shown in *Table 7.4* with ten objects defined over six attributes A_1, A_2, A_3, A_4, A_5 and A_6. The domains for the attributes are respectively, $D_1 = \{a_1,\ a_2,\ a_3\}$, $D_2 = \{b_1,\ b_2\}$, $D_3 = \{c_1,\ c_2,\ c_3,\ c_4\}$, $D_4 = \{d_1,\ d_2,\ d_3\}$, $D_5 = \{e_1,\ e_2\}$ and $D_6 = \{f_1,\ f_2,\ f_3,\ f_4,\ f_5\}$

Clusters C_1, C_2, C_3 and C_4 can be identified in the dataset with $MinmAtt = 6/2 = 3$

$C_1 = \{Olist = \{2, 8\},\ N_{attrib} = 6, D_{attrib} = \{1, 2, 3, 4, 5, 6\},$
$V_{attrib} = \{a_2, b_2, c_4, d_3, e_2, f_2\}\}.$

$C_2 = \{Olist = \{1, 4\},\ N_{attrib} = 5, D_{attrib} = \{2, 3, 4, 5, 6\},$
$V_{attrib} = \{b_2, c_4, d_1, e_2, f_1\}\}.$

$C_3 = \{Olist = \{5, 7\},\ N_{attrib} = 5, D_{attrib} = \{1, 2, 3, 5, 6\},$
$V_{attrib} = \{a_3, b_1, c_2, e_1, f_3\}\}.$

$C_4 = \{Olist = \{3, 9, 10\},\ N_{attrib} = 5, D_{attrib} = \{1, 2, 3, 4, 5\},$
$V_{attrib} = \{a_3, b_1, c_2, d_1, e_1\}\}.$

6.3.2.3 k-point Algorithm

The unsupervised classification algorithm starts with an empty set of clusters. Initially, k objects are selected randomly from the dataset. It reads each object X_i sequentially from dataset, and inserts X_i in an existing cluster based upon the similarity between X_i and a cluster. If the similarity between a cluster and the object does not hold, a new cluster is created with X_i, if X_i is similar with any of the randomly selected k objects for a defined threshold $MinmAtt$ of attributes. Search for a cluster for inserting an object is started from the beginning of the created cluster set until the search is successful. The objects which are neither able to include in any one of the clusters nor create a new cluster based upon the defined $MinmAtt$ are excluded from the clusters. Based upon similarity on profiles, similar clusters are merged into single cluster. The largest cluster is selected to identify its label of normal on basis of the assumption of normal behaviour model. The algorithm is given as Algorithm 8.4.2.

Selection of k

We have analyzed the effect of selection of k value for k-point algorithm using the benchmark dataset *Corrected* KDD and *KDDTest+* of NSL-KDD dataset and real life dataset *Packet level* and *Portscan*. The performance of the proposed method in terms of *PCC* depends on the selection of k value as seen in Figures 6.3. It is dependent on the dataset used for evaluation. However, a most probable range of k for these datasets is shown with vertically drawn dashed lines in Figures 6.3. In our experiments, better results are found with k value in between $k = 10$ and $k = 12$.

6.3.3 Complexity analysis

The k-point algorithm requires one pass through the dataset. Each object need to be compared with existing clusters one after another until it gets inserted in one of the clusters. The similarity computation involves a subset of attributes. Therefore, the clustering process has a complexity $O(ncd)$, where n is the number of objects in dataset, c is the number of clusters, and d is the number of attributes. Each of the created clusters need to be visited for k number of objects for d attributes. Hence, maximum time complexity

5 k-point Algorithm

```
Input: Dataset $D$,Value of $k$;
Output: Largest Cluster, $C_L$;
 1: Select $k$ records randomly from the $D$;
 2: Find number of attributes $M$ of each of $k$ records
 3: $S = \emptyset$; $MinmAtt = M/2$; $T = M$;
 4: if $D \neq \emptyset$ then
 5:     Fetch a record $d$ from $D$;
 6: end if
 7: if $d$ is unselected record and $S \neq \emptyset$ then
 8:     Find a cluster $C_i$ from $S$ ;
 9:     Compute similarity, $simC\,(C_i, d)$ between cluster $C_i$ and object $d$;
10: else
11:     Goto step 5;
12: end if
13: if $simC\,(C_i, d) == 0$ then
14:     Add record $d$ to $C_i$;
15:     Go to step 5;
16: end if
17: Compute similarity, $simO\,(d, k_i)$ between object $d$ and any record $k_i$ from
    random records for $T$ attributes;
18: if $simO\,(d, k_i) == 0$ then
19:     Create a cluster $C_j$ with $Clusprofile = (N_{attrib}, D_{attrib}, V_{attrib})$;
20:     Include $C_j$ to $S$;
21:     Go to step 5;
22: else
23:     $T = T - 1$;
24: end if
25: if $T > MinmAtt$ then
26:     Go to step 17;
27: else
28:     Go to step 5;
29: end if
30: if Number of clusters in $S > 2$ then    ▷ //Merging among the clusters//
31:     if Profiles of $C_x ==$ Profile of $C_y$ then
32:         Merge Oblist of $C_y$ with Oblist of $C_x$;
33:         Update $S$;
34:     end if
35: end if
36: Find the largest cluster $C_L$ by comparing Oblist of clusters from $S$;
37: Stop;
```

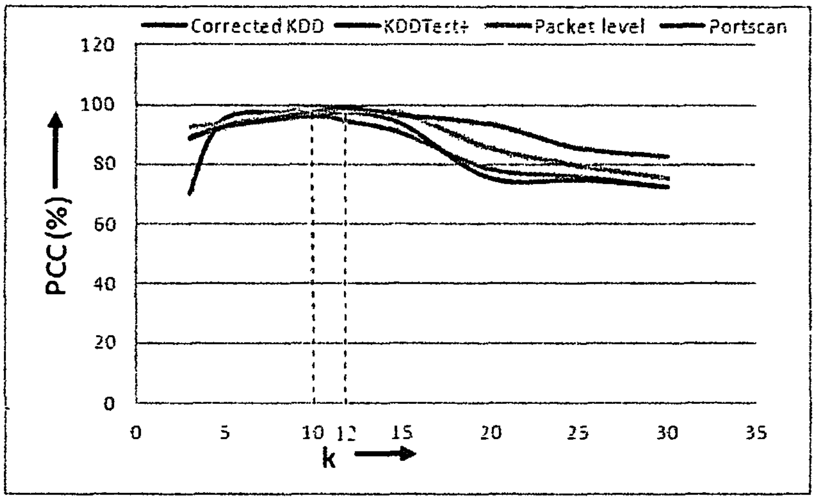


Figure 6.3: k determination for k-point algorithm

Table 6.2: Attack distribution in benchmark intrusion datasets

KDD Cup 1999				NSL-KDD			
Datasets	***Normal***	***Attack***	***Total***	***Datasets***	***Normal***	***Attack***	***Total***
Corrected KDD	60593	250436	311029	*KDDTest*$^+$	9710	12834	22544
10-percent KDD	97278	396743	494021	*KDDTrain*$^+$	67343	58630	125973

of k-point algorithm becomes $O(ncd) + O(kd)$.

6.4 Experimental results

6.4.1 Environment Used

The experiments were carried out in a Intel workstation with configuration of core 2 Quad @2.4GHz, 2 GB RAM, 160GB HDD. The program was executed in Linux environment with *C* compiler.

6.4.2 Datasets Used

The algorithm was tested on two benchmark intrusion datasets, viz., KDD Cup 1999[81] and NSL-KDD[82] datasets. The algorithm was also evaluated with three real life TUIDS[92] intrusion datasets. The description of the datasets are given in *Chapter* 3. The Tables 6.2, 6.3 and 6.4 show the attack distributions of the datasets.

Table 6.3: Attack distribution in TUIDS intrusion datasets

Packet Level				Flow Level			
Datasets	*Normal*	*Attack*	*Total*	*Datasets*	*Normal*	*Attack*	*Total*
Training	71785	50142	121927	*Training*	23120	29723	52843
Testing	47895	38370	86265	*Testing*	16770	23955	40725

Table 6.4: Attack distribution in TUIDS intrusion datasets (continue)

Portscan			
Datasets	*Normal*	*Attack*	*Total*
PortscanTrain	2445	39215	41660
PortscanTest	1300	28615	29915

6.4.3 Results on KDD Cup 1999 Dataset

The confusion matrices for two dataset *Corrected KDD* and 10 *percent KDD* of KDD Cup 1999 Dataset are given in *Table 6.5 and 6.6.* On *Corrected KDD* dataset, the DR for intrusion records is 97.55%, TPR for normal records is 90.01% and FPR for normal records is 2.45%. Similarly, on 10 *percent KDD* dataset, the DR for intrusion records is 95.75%, TPR for normal records is 94.76% and FPR for normal records is 4.25%. The average execution time of classification for *Corrected KDD* and 10 *percent KDD* datasets are 2.65min and 4.21min respectively.

6.4.4 Results on NSL-KDD Dataset

The confusion matrices for two datasets $KDDTrain^{+}$ and $KDDTest^{+}$ of NSL-KDD Dataset are given in *Table 6.7 and 6.8.* On $KDDTrain^{+}$ dataset,

Table 6 5: Confusion matrix for *Corrected KDD* dataset

	Values	Predicted Class				
		Normal	*Attack*	Sum	Recall	1-Prc*
Actual	*Normal*	54540	6053	60593	0.9001	0.1011
class	*Attack*	6135	244301	250436	0.9755	0.0242
	Sum	60675	250354	311029		

Re-substitution error rate=0.0392 PCC=96.08%

Note- *1-Precision

Table 6.6: Confusion matrix for 10 *percent KDD* dataset

	Values	Predicted Class				
		Normal	*Attack*	Sum	Recall	1-Prc*
Actual class	*Normal*	92180	5098	97278	0.9476	0.1545
	Attack	16846	379897	396743	0.9575	0.0132
	Sum	109026	384995	494021		
	Re-substitution error rate=0.0444				PCC=95.56%	

Note- *1-Precision

Table 6.7: Confusion matrix for $KDDTrain^+$ dataset

	Values	Predicted Class				
		Normal	*Attack*	Sum	Recall	1-Prc*
Actual class	*Normal*	63228	4115	67343	0.9389	0.0189
	Attack	1377	57413	58630	0.9792	0.0669
	Sum	64445	61528	125973		
	Re-substitution error rate=0.0423				PCC=95.77%	

Note- *1-Precision

the DR for intrusion records is 97.65%, TPR for normal records is 93.89% and FPR for normal records is 2.35%. Similarly, on $KDDTest^+$ dataset, the DR for intrusion records is 98.88%, TPR for normal records is 96.55% and FPR for normal records is 1.12%. The average execution time of classification for $KDDTrain^+$ and $KDDTest^+$ datasets are 1.07min and 0.07min respectively.

6.4.5 Results on TUIDS Intrusion Dataset

The confusion matrices for two datasets *Packet-level* and *Flow-level* of TUIDS intrusion dataset are given in *Table 6.9 and 6.10.* On *Packet Level* dataset, the DR for intrusion records is 99.29%, TPR for normal records is 98.89% and FPR for normal records is 0.71%. Similarly, on *Flow Level* dataset, the DR for intrusion records is 99.53%, TPR for normal records is 99.11% and FPR for normal records is 0.47%. The average execution time of classification for *Packet-level* and *Flow-level* datasets are 1.04min and 0.45min respectively.

Table 6.8: Confusion matrix for $KDDTest^+$ dataset

	Values	Predicted Class				
		Normal	*Attack*	Sum	Recall	1-Prc*
Actual	*Normal*	9375	335	9710	0.9655	0.0151
class	*Attack*	144	12690	12834	0.9888	0.0257
	Sum	9519	13025	22544		
Re-substitution error rate=0.0212					PCC=97 88%	

Note- *1-Precision

Table 6.9: Confusion matrix for *Packet level* dataset

	Values	Predicted Class				
		Normal	*Attack*	Sum	Recall	1-Prc*
Actual	*Normal*	70981	804	71785	0.9888	0.0050
class	*Attack*	357	49785	50142	0.9929	0.0159
	Sum	71338	50589	121927		
Re-substitution error rate=0.0095					PCC=99.05%	

Note- *1-Precision

Table 6.10: Confusion matrix for *Flow level* dataset

	Values	Predicted Class				
		Normal	*Attack*	Sum	Recall	1-Prc*
Actual	*Normal*	22914	206	23120	0.9911	0.0061
class	*Attack*	140	29583	29723	0.9953	0.0065
	Sum	23054	29789	52843		
Re-substitution error rate=0.0057					PCC=99.35%	

Note- *1-Precision

Table 6.11: Confusion matrix for *PortscanTrain* dataset

Values		Predicted Class				
		Normal	*Attack*	Sum	Recall	1-Prc*
Actual	*Normal*	2412	33	2445	0.9868	0.1165
class	*Attack*	318	38897	39215	0.9919	0.0009
	Sum	2730	38930	41660		

Re-substitution error.rate=0.0084 PCC=99.16%

Note- *1-Precision

Table 6.12: Experimental Results of k-point Algorithm

Data sets	***Total***	***Attacks***	***Normal***	***DR (%)***	***TPR (%)***	***FPR (%)***
Corrected KDD	311029	250436	60593	97.55	90.01	2.45
10 *Percent KDD*	494021	396743	97278	95.75	94.76	4.25
KDDTrain$^+$	125973	58630	67343	97.65	93.89	2.35
KDDTest$^+$	22544	12834	9710	98.88	96.55	1.12
Packet Level	121927	50142	71785	99.29	98.89	1.59
Flow Level	52843	29723	23120	99.53	99.11	0.65
PortscanTrain	41660	39215	2445	99.19	98.68	0.09

6.4.6 Results on TUIDS Portscan Dataset

The confusion matrix for dataset *PortscanTrain* of TUIDS intrusion dataset is given in Table 6.11. On *PortscanTrain* dataset, the *DR* (i.e., Recall) for intrusion records is 99.19%, for normal records is 98.68%. The average execution time of classification for *PortscanTrain* dataset is 0.35min.

6.4.7 Comparison of Results

The summarized results over the three distinguished datasets is presented in *Table 6.12* for detection rate (*DR*) on intrusive records, *TPR* and *FPR* over *normal* records for the datasets.

Table 6.13: Comparison with Unsupervised Techniques on *Corrected KDD*

Algorithm	*Detection Rate (%)*
fpMAFIA[66]	86.70
K-NN[197]	89.95
Fixed width clustering[197]	94.00
Modified Clustering-TV[196]	97.30
k-point algorithm	97.55

6.4.8 Performance Comparisons

The performance of k-point algorithm is compared with other four unsupervised anomaly-based intrusion detection algorithms: fpMAFIA[66], K-NN[197], Fixed width clustering[197] and Modified Clustering-TV[196]. The comparison results of performance over *Corrected KDD* dataset for all these algorithms are shown in *Table 6.13.* In the performance comparison, detection rate for intrusion instances by k-point algorithm is maximum .

6.4.9 Discussion

In this chapter we provide a clustering algorithm, k-point and applied it in unsupervised anomaly-based network intrusion detection. We developed the clustering method by building normal behaviour model from unlabeled dataset. We evaluated the anomaly detection approach by applying it on two benchmark intrusion datasets and three real life private intrusion datasets. The performances of our method is compared with existing unsupervised network anomaly detection methods. The method is capable to establish its effectiveness in network anomaly detection as indicated by the evaluation results on real life network intrusion dataset and the benchmark intrusion datasets.

The method given in *Section 5.3.2.1 and 5.3.3* is different from the k-point method. The k-point method finds the exact matching of certain number of attributes while the other method compares the total value of matching of attributes between two records. Also, both the methods use different proximity measures. The k-point method has comparatively lower performance than the supervised methods due to lack of training dataset.

Outlier detection methods are applicable for known as well as unknown attack identification in network anomaly detection. In the next *Chapter 7*, an outlier detection method is introduced for network anomaly detection.

CHAPTER 7

Anomaly Detection Using Outlier Approach

Contents

Outlier mining is a widely used method for finding exceptional or rarely occurring instances in different real world scenarios. In this chapter we propose an outlier mining method in network anomaly detection by identifying rare occurrence network traffic as attack.

7.1 Introduction

Outliers identification refers to the problem of finding data points that are very different from the rest of the data based on appropriate metrics. Such data points often contain useful information regarding unusual behaviour of a system described by the data. These anomalous data points are usually called outliers. Outlier detection has been used widely in finding anomalous activity in telecommunication, credit card fraud detection, detecting symptoms of new diseases and novel network attack detection.

Certain piece of information are usually required by an outlier detection method to work. One such piece of information is a labeled training data set that can be used with techniques from machine learning[201] and statistical learning theory[168]. An explicit predictive model is built based on the training dataset for detection. The associated label with an instance of data refers to the data instance either as normal or intrusion class. The outlier detection methods can be supervised or unsupervised based on the extent to which these labels are utilized or available. A supervised outlier detection method has available to it a labeled training dataset to build a predictive model for both normal and intrusion classes. An unsupervised outlier detection method is used when a labeled training dataset is unavailable. The methods in this category make assumptions about the data. Several methods assume that normal instances are more frequent than intrusion instances.

Intrusion detection (ID) is an important component of any infrastructure protection mechanism. It is a part of any security management system for computers and networks. An intrusion detection system (IDS) gathers and analyzes information from various areas within a computer or a network to identify possible security breaches, which include both types of intrusions - internal and external. A misuse intrusion detection approach uses information about known attacks and detects intrusions based on matches with existing attack patterns or signatures. On the other hand, an anomaly detection approach learns the normal behaviour of the system or the network it monitors and reports when the monitored behaviour deviates significantly from the normal profile. There exists various IDSs that are based on misuse as well as anomaly detection. Examples include Bro[202], Snort[144], ADAM[22], NFIDS[146], etc. Several anomaly detection approaches are discussed in[178].

7.1.1 Outlier Detection in Network Anomaly Detection

Network anomaly detection involves the collection of network traffic data relating to the user behaviour over a period of time, and then applying tests to the gathered data to determine whether that behaviour is of legitimate user one or not Anomaly detection has the advantage of detection possibility of new attacks which the system has never seen before and they deviate from normal behaviour.

The major challenge for outlier detection in the context of intrusion detection is to handle huge volume of mixed type- numerical and categorical data. So, outlier detection schemes need to be computationally efficient to handle these large sized inputs in a faster manner. An outlier can be an observation that is distinctly different or is at a position of abnormal distance from other values in the dataset. Detection of abnormal behaviour can be based on relevant features extracted from network traces, packet or flow data. An intrusion can be detected by finding an outlier whose features are distinctly different from the rest of the data. Outliers can often be individuals or groups of clients exhibiting behaviour outside the range of what is considered as *normal*. In order to apply outlier detection to anomaly based network intrusion detection, we assume[8] the followings: (i) The majority of the network connections is normal traffic. Only a small amount of traffic is malicious and (ii) Attack traffic is statistically different from normal traffic.

However, in a real network scenario, these assumptions may not be always true. In dealing with DDoS (Distributed Denial of Service)[203] or bursty attack[204] detection, the anomalous traffic is actually more frequent than the normal traffic.

7.2 Fundamentals of Outlier Detection

Outlier detection searches for objects that do not obey rules and expectations valid for the major part of the data. The detection of an outlier object may be evidence that there are new tendencies in the data. Although, outliers are considered noise or error, they may have important information. The detection of outliers often depends on the applied methods and hidden assumptions regarding data structures used.

7.2.1 Problem Formulation

Outliers refers to the data points or instances that are very different from the rest of the data based on appropriate metrics in dataset. Such data points often contain useful information regarding unusual behaviour of a system described by the data.

With this consideration in view, the outlier can be stated as follows: given a set of data elements $p_1, p_2, \ldots, p_n$ in a dataset D, a classifier C based on a metric M classifies the elements into N groups, then the classified group(s) $G \in N$ is outlier if its number of containing elements are less than a predefined threshold value q.

7.2.2 Outlier Detection Approaches

An outlier is a data point which is very different from the rest of the data set based on some measure. Such point often contains useful information on abnormal behaviour of the system described by data. Depending on the approaches used in outlier detection, the methodologies can be broadly classified[61] as- distance-based, density-based and soft computing based.

7.2.2.1 Distance-based Outlier Detection

Distance-based methods for outlier detection are based on the calculation of distances among objects in the data with clear geometric interpretation. We can calculate a so-called outlier factor as a function $F\ :\ x \rightarrow\ R$ to quantitatively characterize an outlier[205]. The function F depends on the distance between the given object x and other objects R in the dataset being analysed.

In LOADED[113], Ghoting et.al. introduce a distance based outlier detection method for mixed attribute data. Here, data points are considered linked if they are similar to each other for each attribute pair. A density-based outlier detection method estimates the density of the neighbourhood of each data instance. An instance that lies in a neighbourhood with low density is declared to be an outlier while an instance that lies in a dense neighbourhood is declared to be normal. Breunig et al.[114] computed a local outlier factor (LOF) for each object in the dataset. The outlier factor quantifies outlyingness of an object. In ODMAD[79], an anomaly score is computed for each data point taking into consideration the irregularity of the categorical values, the continuous values, and the relationship between the two spaces in the dataset. The data points with similarity close to '0' are more likely to be outliers.

7.2.2.2 Density-based Outlier Detection

Density-based methods estimate the density distribution of the input space and then identify outliers as those lying in regions of low density[79]. *Density-based* outlier detection techniques estimate the density of the neighbourhood of each data instance. An instance that lies in a neighbourhood with low density is declared to be an outlier while an instance that lies in a dense neighbourhood is declared to be normal.

Peng Yang and Biao Huang[206] present a modified density based outlier mining algorithm. In this method, for every object in a dataset D does not need to judge whether there are core objects within the ε-neighbourhood of it or not. For given object $o \in D$, the ε-neighbourhood of o is the region with center o and radius ε. The object set within ε-neighbourhood of o is denoted as $o_{\varepsilon\text{-}set}$. If number of object within the ε-neighbourhood of data object C is more than m which is a user defined parameter, C is the core object. Let, D is a dataset and C is the core object, where $C \in D$ and $o \in D$. Given a number $m, \forall C \in D$, if o is not within the ε-neighbourhood of C and $|o_{\varepsilon\text{-}set}| \leq m$, o is the outlier with respect to ε and m. Apart from this, the module information of data object is introduced in this algorithm which reduces large number of computations in finding all outliers.

Jin et al., in[207] focus on mining outliers, called local outliers, that have density distribution significantly different from their neighbourhood. In estimating density distribution in data objects, this method considers both neighbours and reverse neighbours of an object.

7.2.2.3 Outlier Detection based on Soft computing Approaches

Soft computing based outliers are inspired by soft computing (*referred to subsection* 4.6) approaches viz., *RMF (rough membership function)*-based outliers[130,208]. A RMF is defined as follows.

Let $IS = (U, A, V, f)$ be an information system, $X \subseteq U$ and $X \neq \emptyset$. U is a non-empty finite set of objects, A a set of attributes, V the union of attribute domains, and $f : U \times A \rightarrow V$ a function such that for any $X \in U$ and $a \in A$, $f(x, a) \in V_a$. Let ν be a given threshold value. For any $x \in X$, if $ROF_X(x) > \nu$, x is called a *rough membership function* (*RMF*)*-based outlier* with respect to X in IS, where $ROF_X(x)$ is the rough outlier factor of x with respect to X in IS The *rough outlier factor* is defined as

$$ROF_X(x) = 1 - \frac{\sum_{j=1}^{m} \left(\mu_X^{A_j}(x) \times |A_j|\right) + \sum_{j=1}^{m} \left(\mu_X^{\{a_j\}}(x) \times W_X^{\{a_j\}}(x)\right)}{2 \times |A|^2} \tag{7.1}$$

where $A = \{a_1, a_2, \ldots, a_m\}$. $\mu_X^{A_j}(x)$ and $\mu_X^{\{a_j\}}(x)$ are RMFs for every attribute subset $A_j \subseteq A$ and singleton subset $\{a_j\}$ of A, $1 \leq j \leq m$. For every singleton subset $\{a_j\}$, $W_X^{\{a_j\}} : X \rightarrow (0, 1]$ is a weight function such that for any $x \in X$, $W_X^{\{a_j\}}(x) = \sqrt{(|\,[x]_{\{a_j\}}\,|)/(|U|)}$. $[x]_{\{a_j\}} \doteq \{u \in U : f(u, a_j) = f(x, a_j)\}$ denotes the indiscernibility class of relation IND$(\{a_j\})$ that contains element x.

The RMF is $\mu_X^B :\rightarrow (0, 1]$ such that for any $x \in X$

$$\mu_X^B(x) = \frac{|[x]_B \cap X|}{|[x]_B|} \tag{7.2}$$

where $[x]_B = \{u \in U \,.\, \forall_a \in B(f(u, a) = f(x, a))\}$ and $B \subseteq A$ denotes the indiscernibility class of relation IND(B) that contains element x.

Rough sets are used in classification system, where we do not have complete knowledge of the system[131]. In any classification task, the aim is to form various classes where each class contains objects that are not noticeably different. These indiscernible or indistinguishable objects can be viewed as ba-

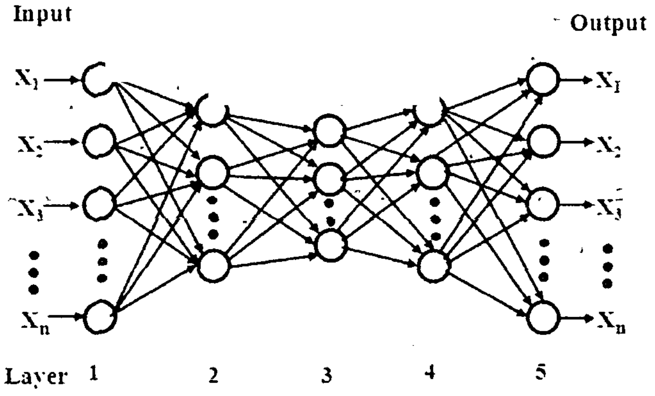


Figure 7.1: A schematic view of fully connected RNN

sic building blocks (concepts) used to build a knowledge base about the real world. This kind of uncertainty is referred to as rough uncertainty. Rough uncertainty is formulated in terms of rough sets.

In most of machine learning or soft computing based outlier detection methods- Rough membership function (RMF) (refer to subsection 4.6.2)[208], Replicator Neural Networks (RNN)[209], Gaussian Mixture Model (GMM) (refer to subsection 4.5.4)[210], outlier deciding scores are estimated in the data objects to identify outlier data objects

Hawkins et al. present an outlier detection approach on basis of Replicator Neural Networks (RNN)[209]. RNN is a feed-forward multi-layer approach. It consists of three hidden layers which are framed in between an input and output layer. The function of the RNN is to reproduce the input data pattern at the output layer with minimized error through training. Both input and output layers have n units,. corresponding to the n features of the training data. The number of units in the three hidden layers are experimentally chosen to minimize the average reconstruction error over all training patterns. Figure 7.1 shows a schematic view of the fully connected RNN. The output of unit i of layer k is calculated by the activation function $S_k(I_{ki})$, where I_{ki}, is the weighted sum of the inputs to the unit and defined as: $I_{ki} = \sum_{j=0}^{L_{k-1}} w_{kij} Z_{(k-1)j}$. Z_{kj} is the output from jth unit of the kth layer. The activation function for the two outer hidden layers ($k = 2, 4$) is then: $S_k(I_{ki}) = \tanh(a_k I_{ki})$, where a_k is a tuning parameter which is set to 1 usually.

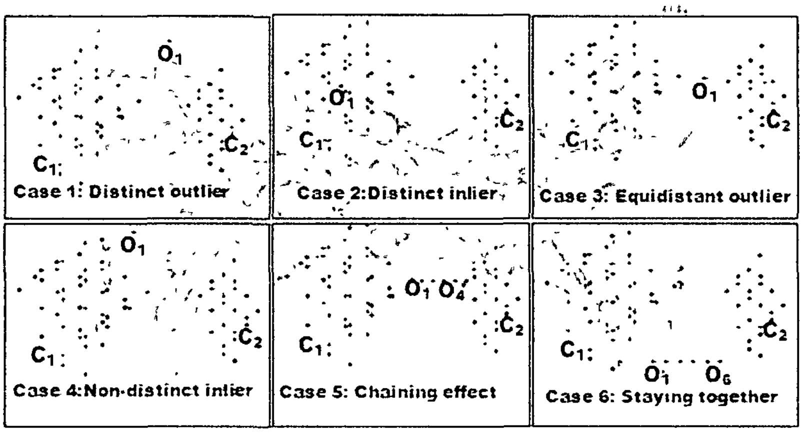


Figure 7.2: Six cases of outlier

RNN is trained from a sampled dataset to build a model which predicts the given data. This model is used to develop a score of outlyingness called outlier factor where the trained RNN is applied to the entire dataset to give a quantitative measure of the outlyingness based on the reconstruction error. This approach takes the view of expressing itself without relying on too many assumptions. This approach identifies cluster labels for each data record. Sometimes, outliers are found concentrated in a single cluster. Without using class labels, RNN has capability to identify outliers considering small classes with high accuracy.

7.2.3 Discussion

An outlier is an observation that deviates from other observations such that it creates suspicion as if it is generated by a different mechanism. Detected outliers indicate the behaviours outside the range of 'normal' one. Three popular outlier detection methods: *distance based*, *density based* and *soft computing based* are discussed.

Based on our study to evaluate the effectiveness of outlier detection methods, we can consider six cases as shown in Figure 7.2 over the synthetic data set. In these figures, O_i is an object and C_i is cluster of objects. Following are our observations:

1. A *distinct outlier* object, (*case 1* in Figure 7.2) is one that cannot be included in any clusters.
2. A *distinct inlier* object (*case 2*) is inside of a cluster.
3. An *equidistant outlier* (*case 3*) is the object which is at equal distance from the clusters.
4. *Non-distinct inlier* (*case 4*) is that object located near the border of a cluster.
5. The *chaining effect* (*case 5*) represents the objects which are situated in a straight line among the clusters.
6. The *staying together* (*case 6*) effect represents the objects which are outliers on a straight line.
7. Most existing outlier detection methods cannot handle all the cases as reported in Figure 7.2.

7.2.4 Contribution

Most existing methods for outlier detection in the literature are based on density estimation methods or nearest-neighbour methods[114,211]. In this chapter, we introduce an effective method for outlier identification using symmetric neighbourhood relationships[207], considering nearest neighbours as well as reverse of the nearest neighbours. We present experimental results which show the ability to find anomalies when detecting rare intrusion instances in the KDD Cup 1999[81] and NSL-KDD[82] datasets. Results from outlier detection using the method for various UCI ML Repository[212] datasets and our real life intrusion datasets TUIDS[92] are also provided.

7.3 Proposed Method

We have developed an outlier mining method based on symmetric neighbourhood relationship[207] for mixed type data. For each object of data set a forward neighbour outlier factor is estimated by finding nearest neighbour set and forward nearest neighbour set of the data objects to identify outliers.

Nearest Neighbour Set of k objects (NNk)

In a dataset $D = \{d_1, d_2, \ldots, d_n\}$ of n objects, d_i and d_j are two arbitrary objects in D. We use Euclidean distance to evaluate the distance between objects d_i and d_j which is denoted as $dist(d_i, d_j)$.

Definition 1- Nearest Neighbour Set of k objects of object p is the set of k nearest neighbour objects of p or $NNk(p)$ where $k > 0$. In dataset D of $|D|$ objects

$$|NNk(p)| = k. \tag{7.3}$$

if $\forall p \notin NNk(p)$ *and* $dist(o,p) < dist(\acute{o},p)$,where o and $\acute{o}$ are k-th and $(k+1)$-th nearest neighbour to p respectively.

Forward Nearest Neighbour Set of k objects ($FNNk$)

Definition 2- Forward Nearest Neighbour Set of k objects of object p is the set of objects whose NNk contains p, denoted as $FNNk(p)$. In dataset D of $|D|$ objects where p and q are arbitrary objects, $FNNk(p)$ is defined as,

$$FNNk(p) = \{q \in D' | \, p \in NNk(q) \text{ and } p \neq q\}. \tag{7.4}$$

Forward Neighbour Outlier Factor of k objects ($FNOFk$)

Definition 3- Forward Neighbour Outlier Factor of k objects for an object p is the ratio of remaining number of objects of $FNNk(p)$ of dataset D except object p to the number of dataset objects except object p, denoted as $FNOFk(p)$. In dataset D of objects $|D|$, $FNOFk(p)$ is defined as,

$$FNOFk(p) = \frac{|D| - |FNNk(p)| - 1}{|D| - 1} = 1 - \frac{|FNNk(p)|}{|D| - 1}. \tag{7.5}$$

Example 1: A sample 2-D dataset is given in *Table 7.1*. The dataset have six objects p,q_1,q_2,q_3,q_4 and q_5. The dataset is plotted in *Fig. 7.3*.

For neighbourhood size $k = 3$, the NNk and $FNNk$ will be as follows.

$NNk(p) = \{q_1, q_2, q_3\}$, $FNNk(p) = \{q_1, q_2, q_3, q_4\}$,

Table 7.1: A sample dataset

Objects	x-ordinate	y-ordinate
p	19	8
q_1	18.5	12
q_2	22	9
q_3	23.5	9
q_4	14	15
q_5	22	15

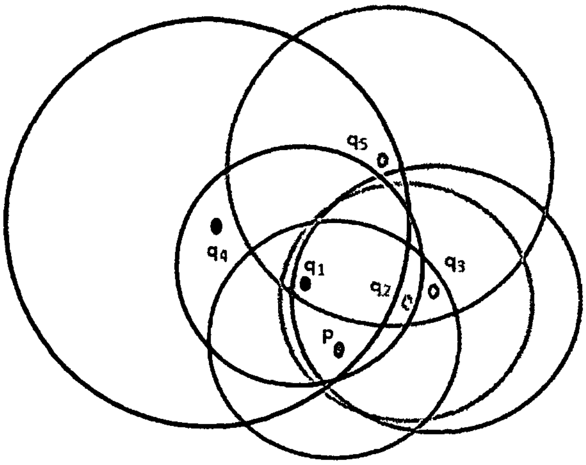


Figure 7.3: Example 1- dataset plot

$NNk(q_1) = \{p, q_2, q_4\}$, $FNNk(q_1) = \{p, q_2, q_3, q_4, q_5\}$,
$NNk(q_2) = \{p, q_1, q_3\}$, $FNNk(q_2) = \{p, q_1, q_3, q_5\}$,
$NNk(q_3) = \{p, q_1, q_2\}$, $FNNk(q_3) = \{p, q_2, q_5\}$,
$NNk(q_4) = \{p, q_1, q_5\}$, $FNNk(q_4) = \{q_1\}$,
$NNk(q_5) = \{q_1, q_2, q_3\}$, $FNNk(q_5) = \{q_4\}$.

The number of objects of NNk and $FNNk$ for $k = 3$ are as follows.
$|NNk(p)| = 3$, $|FNNk(p)| = 4$,
$|NNk(q_1)| = 3$, $|FNNk(q_1)| = 5$,
$|NNk(q_2)| = 3$, $|FNNk(q_2)| = 4$,
$|NNk(q_3)| = 3$, $|FNNk(q_3)| = 3$,
$|NNk(q_4)| = 3$, $|FNNk(q_4)| = 1$,
$|NNk(q_5)| = 3$, $|FNNk(q_5)| = 1$.

The outlier factor $FNOFk$ for $k = 3$ and $|D|$=6 are as follows.

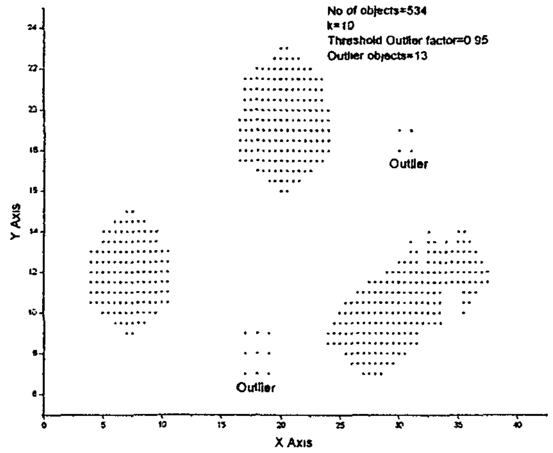


Figure 7.4: Example 2- dataset plot

$FNOFk(p) = (1 - 4/5) = 0.20,$
$FNOFk(q_1) = (1 - 5/5) = 0.0,$
$FNOFk(q_2) = (1 - 4/5) = 0.20,$
$FNOFk(q_3) = (1 - 3/5) = 0.40,$
$FNOFk(q_4) = (1 - 1/5) = 0.80,$
$FNOFk(q_5) = (1 - 1/5) = 0.80.$
The outliers are found for threshold $M_{th} \geq 0.80$ and $k = 3$ as follows.
$FNOFk(q_4) = 0.80$ and $FNOFk(q_5) = 0.80$.

Example 2: A plot of sample 2-D dataset is given in *Fig. 7.4* for data objects 534. Here, 17 data objects are identified as outlier for value of $k = 20$ and threshold $FNOF$, $M_{th} = 0.95$.

Outlier Detection Algorithm

The outlier detection algorithm and two associated functions- $getFNOFk(D, k)$ and $getNNk(D, p, k)$ are given in Algorithm 6. The algorithm has of two functions: $getNNk()$ and $getFNOF()$.

Algorithm 6 Outlier Detection Algorithm

```
Input: Dataset $D$, Threshold $M_{th}$, $Neighbour\_size$ $k$ , Object $p$;
Output: Outlier List $L$;
$X = getFNOFk(D, k)$;
$\forall_{t \in X}$
if $t \geq M_{th}$ then
    Add $t$ to $L$;
end if

function $getFNOFk(D, k)$
    while $|D| \neq \emptyset$ do
        $\forall_{p \in D}$ $S = getNNk(D, p, k)$;
    end while
    $\forall_{q \in S}$ $T = getNNK(D, q, k)$;
    if $p \in T$ then
        Add $q$ to list of $FNNk(p)$;
        $|FNNk(p)| = |FNNk(p)|+1$;
    end if
    Compute $\forall_{p \in D} FNOFk(p) = \left\{1 - \frac{|FNNk(p)|}{|D|-1}\right\}$;
    return $FNOFk(p)$;
end function

function $getNNk(D, p, k)$
    if $|D| \neq \emptyset$ then
        $\forall_{q;p \neq q,p,q \in D}$ Compute $dist(p, q)$;
    end if
    $\forall_q$ Sort $dist(p, q)$;
    Add $k$ shortest distant objects from $p$ to $NNk(p)$;
    return $NNk(p)$;
end function
```

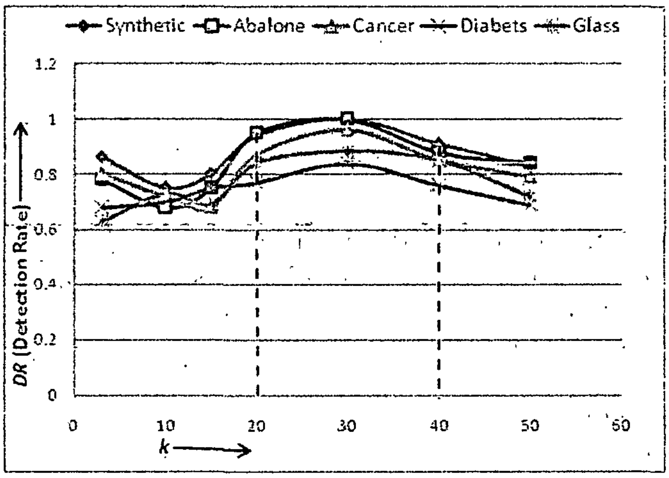


Figure 7.5: k value determination

7.3.1 Complexity analysis

The outlier detection algorithm basically comprises of two functions. *getNNk*() and *getFNOF*(). The complexity of function *getNNk*(), is due to the distance computation of n objects and sorting of n object distances (n log n, as we used Quicksort algorithm). Thus the time complexity of this function is $O(n + n\ log\ n)$. Again, the complexity of function *getFNOF*() is due to searching of $n \times k$ objects, computation of outlier factor for n objects and searching of n objects for user defined threshold value. Thus, the time complexity of this function is $O(n \times k \times (n + n\ log\ n) + 2n)$. Hence, time complexity of the outlier algorithm is $O(n \times k \times (n + n\ log\ n) + 2n) \cong O(n^2)$.

7.3.2 Dependency on Parameters

We have analyzed the effect of neighbourhood value k and threshold *M_thresh*, using synthetic as well as real life datasets (i.e., abalon, breast-cancer, diabetes and glass). The performance of the proposed method in terms of *DR* largely depends on the selection of k and *M_thresh*, as seen in *Figure 7.5-7.6*. It is dependent on the dataset used for evaluation. However, a most probable range of k and *M_thresh* for these datasets are shown with vertically drawn dashed lines in the *Figure 7.5-7.6*. In our experiments, better

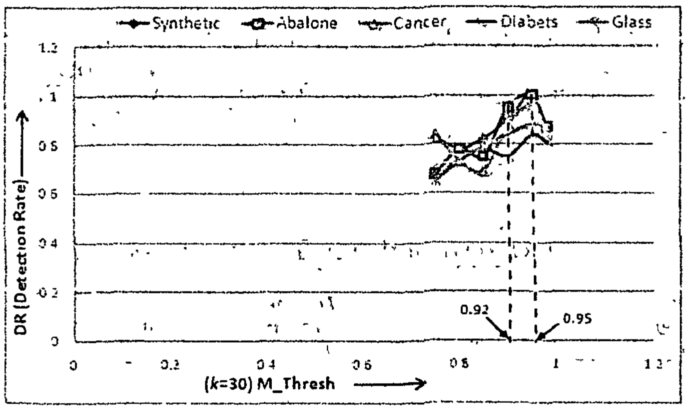


Figure 7.6: M-thresh determination

results are found with k values in the range of (0.20 - 0.40) and M_thresh value in the range of (0.92 - 0.95). Our experiments are evaluated with $k = 30$ and $M\_thresh = 0.93$.

7.4 Experimental Results

In this section, we perform a comprehensive experimental evaluation of the efficiency and the effectiveness of our outlier detection method.

7.4.1 Environment Used

All necessary experiments were carried out on an Intel workstation with Intel core 2 Quad @2.4GHz, 2 GB RAM, 160GB HDD. The programs were developed in *c* in a Linux environment.

7.4.2 Dataset Used

The method was evaluated with various UCI ML Repository dataset[212], a synthetic dataset, KDD Cup 1999 dataset[81], NSL-KDD[82] datasets and our real life intrusion datasets TUIDS[92]. The description of datasets KDD Cup 1999, NSL-KDD and TUIDS are given in *Chapter* 3. The Tables 7.2, 8.5 and 7.4 show the attack distributions of these datasets. The description of UCI ML Repository dataset and synthetic dataset are given below.

Table 7.2: Attack distribution in benchmark intrusion datasets

KDD Cup 1999				NSL-KDD			
Datasets	*Normal*	*Attack*	*Total*	*Datasets*	*Normal*	*Attack*	*Total*
Corrected KDD	60593	250436	311029	$KDDTest^{+}$	9710	12834	22544
10-percent KDD	97278	396743	494021	$KDDTrain^{+}$	67343	58630	125973

Table 7.3: Attack distribution in TUIDS intrusion datasets

Packet Level				Flow Level			
Datasets	*Normal*	*Attack*	*Total*	*Datasets*	*Normal*	*Attack*	*Total*
Training	71785	50142	121927	*Training*	23120	29723	52843
Testing	47895	38370	86265	*Testing*	16770	23955	40725

7.4.2.1 UCI ML Repository and Synthetic dataset

To establish the effectiveness of the proposed outlier method, initially we experimented with several real life benchmark datasets of different dimensionality and number of isolated objects. A major reason of choosing these datasets is to compare our results with the other competing outlier algorithms.

Our method was tested on several commonly used datasets from the UCI ML Repository dataset[212]. We considered eighteen different datasets and a synthetic dataset for experimentation. The descriptions of datasets are given in Table 7.5. The synthetic dataset is a 2-dimensional dataset which was generated with different outlier cases. The other datasets are- abalone, breast cancer, diabetes, glass, heart, hepatitis, horse, ionosphere, iris, led7, pima, poker hand, sonar, shuttle, vehicle, waveform, wine and zoo.

7.4.3 Data preprocessing

The datasets contain categorical and numerical values. The dataset is preprocessed to convert it into an equivalent numeric dataset. The categorical attributes of dataset are replaced with numeric value. The continuous valued attributes of dataset are converted to discrete numbers by taking logarithm

Table 7.4: Attack distribution in TUIDS intrusion datasets (continue)

Portscan			
Datasets	*Normal*	*Attack*	*Total*
PortscanTrain	2445	39215	41660
PortscanTest	1300	28615	29915

Table 7.5: Characteristics of Synthetic and UCI ML Repository datasets

Sl No.	Datasets	Dimension	Instances	Classes	Instances of Outlier
1	Synthetic	2	2000	10	80
2	Abalone	8	4177	29	24
3	Breast cancer	10	699	2	39
4	Diabetes	8	768	2	43
5	Glass	10	214	6	9
6	Heart	13	270	2	26
7	Hepatitis	20	155	3	32
8	Horse	28	368	2	23
9	Ionosphere	34	351	3	59
10	Iris	4	150	4	27
11	Led7	7	3200	9	248
12	Pima	8	768	2	15
13	Poker Hand	10	25010	10	16
14	Sonar	60	208	2	90
15	Shuttle	9	14500	3	13
16	Vehicle	18	846	4	42
17	Waveform	21	5000	3	87
18	Wine	13	178	3	45
19	Zoo	18	101	7	17

to the base 2 and then converting to integer. This is done for each attribute value, z using the computation: *if* $(z > 2)$ $z = int\ (log_2\ (z)) + 1$. Before taking logarithm, the attributes which take fractional values in the range [0, 1] are multiplied by 100 so that they take values in the range [0, 100]. Nominal valued attributes are mapped to discrete numeric codes. The class label attribute is removed from the dataset and stored separately in a different file The class labels are used for training and evaluating the detection performance of the algorithm.

7.4.4 Results on UCI ML Repository Dataset

The proposed method was evaluated using our own 2-dimensional synthetic dataset consisting of 2000 objects and 4% of outlier objects as given in *Table 7.5*. The results of synthetic dataset and other UCI Machine Repository datasets are given in terms of *detection rate* (DR) and *false positive rate* (FPR) in *Table 7.6*. The results are compared with the results of other outlier finding methods LOF[213] and ORCA[214]. Our method has produced better result compared to the other methods. The comparison results is given in *Table 7.6*.

Table 7.6: Experimental Results of Synthetic and UCI ML Repository datasets

Datasets	Measure	LOF[213]	ORCA[214]	Our Method
Synthetic	DR	0.7900	0.8400	**1.0000**
	FPR	0.0239	0.0186	0.0000
Abalone	DR	0.8702	0.8591	**1.0000**
	FPR	0.0443	0.0480	0.0150
Breast Cancer	DR	0.8543	0.8309	**0.9621**
	FPR	0.0377	0.0465	0.0349
Diabetes	DR	0.6813	0.5945	**0.8391**
	FPR	0.0585	0.0458	0.0178
Glass	DR	0.8713	0.8488	**0.8826**
	FPR	0.0360	0.0347	0.0149
Heart	DR	0.9008	0.8869	**0.9828**
	FPR	0.0045	0.0207	0.0211
Hepatitis	DR	0.8902	0.7621	**0.9822**
	FPR	0.0257	0.0399	0.0011
Horse	DR	0.8112	0.8922	0.8705
	FPR	0.0299	0.0405	0.0119
Ionosphere	DR	0.8208	0.8988	**0.9783**
	FPR	0.0377	0.0412	0.0208
Iris	DR	0.7911	0.8933	0.8900
	FPR	0.0311	0.0390	0.0021
Led7	DR	0.2417	0.7510	**0.8819**
	FPR	0.1565	0.0399	0.0048
Pima	DR	0.9443	0.9141	**1.0100**
	FPR	0.0023	0.0221	0.0110
Poker Hand	DR	0.9720	0.9478	0.8911
	FPR	0.0213	0.0290	0.0105
Sonar	DR	0.9800	0.8977	0.9786
	FPR	0.0221	0.0490	0.0210
Shuttle	DR	0.8421	0.7192	0.8230
	FPR	0.0510	0.0341	0.0203
Vehicle	DR	0.3195	0.3919	**0.6428**
	FPR	0.0688	0.0751	0.0335
Waveform	DR	0.8913	0.8587	0.8209
	FPR	0.0460	0.0405	0.0350
Wine	DR	0.9433	0.9322	**0.9810**
	FPR	0.0366	0.0279	0.0100
Zoo	DR	0.8238	0.8823	**0.9421**
	FPR	0.1924	0.1109	0.0338

Table 7.7: Extracted Feature distribution in KDD Cup datasets

Class	*Total Feature*	*Selected Features*
Normal	6	5,23,3,6,35,1
DoS	8	5,23,6,2,24,41,36,3
Probe	13	40,5,33,23,28,3,41,35,27,32,12,24,28
U2R	10	5,1,3,24,23,2,33,6,32,4,14,21
R2L	15	3,13,22,23,10,5,35,24,6,33,37,32,1,37,39

Table 7.8: Experimental Results of KDD Cup 1999 datasets

Dataset		*DR*					*FPR*
		DoS	*R2L*	*U2R*	*Probe*	*All Attacks*	
Corrected KDD	k=3	22.1	75.8	71.3	29.6	25.74	1.3
	k=10	24.3	78.5	74.3	31.2	27.97	1 1
	k=20	32 1	89.0	84 3	39.8	35 96	0.9
	k=30	31.33	55.21	81.12	35.16	32.97	1.03
10 percent KDD	k=3	26 3	81 8	73 4	34.6	26 54	1 2
	k=10	27.3	75.8	82 3	27.50	48 0	1.5
	k=20	30 1	87.0	85.2	41 8	30.39	1.3
	k=30	39.32	63.14	87.25	30.35	39.30	1.12

7.4.5 Results on KDD Cup 1999 Datasets

The proposed method was evaluated using two datasets *Corrected KDD* and 10 *percent KDD* of KDD Cup 1999 datasets. The objective of the experiment is how accurately the method can classify the rare categories of attack data- *R2L* and *U2R*. In high dimensional data, feature selection using mutual information (MI)[84] is a widely accepted approach. The list of relevant features of attacks in KDD Cup datasets is given in *Table 7.7*. The relevant features are considered in outlier detection in KDD Cup datasets. The results using the measures *DR* and *FPR* are reported in *Table 7.8*. The average execution time of classification for *Corrected KDD* and 10 *percent KDD* datasets are 1.11min and 1.74min respectively.

Table 7.9: Experimental Results of NSL-KDD datasets

Dataset		*DR*					*FPR*
		DoS	*R2L*	*U2R*	*Probe*	*All Attacks*	
	k=3	25.4	79 5	73.2	34.1	24.48	1.9
	k=10	21.9	82.6	77.4	34.1	26.89	1.5
$KDDTest^{+}$	k=20	33.2	87.7	89.2	45.5	36.91	0.98
	k=30	37.23	63.21	82.12	37.17	36.89	1.43
	k=3	29.5	89 7	75.9	38.2	32.34	1.8
	k=10	31.4	78.9	87.6	29.60	54.0	1.6
$KDDTrain^{+}$	k=20	36.3	89.2	84.3	47.4	35.52	1.8
	k=30	42.52	69.64	88.65	37.45	46.20	1.42

7.4.6 Results on NSL-KDD Datasets

The proposed method was evaluated using two datasets $KDDTrain^{+}$ and $KDDTest^{+}$ of NSL-KDD datasets. NSL-KDD is the modified dataset of KDD Cup 1999 datasets. A list of relevant features of attacks using mutual information (MI)[84] in NSL-KDD datasets is also given in *Table 7.7*. The relevant features are considered in outlier detection in the datasets. The results using the measures DR and FPR are reported in Table 7.9. DR for all attacks varies in the range of 24.48-36.91 (%) for $KDDTest^{+}$ dataset and 32.34-54.0 (%) for $KDDTrain^{+}$ dataset respectively in Table 7.9. Similarly, FPR exhibits in the range of 0.98-1.9 (%) for $KDDTest^{+}$ dataset and 1.42-1.8 (%) for $KDDTrain^{+}$ dataset respectively in Table 7.9. The average execution time of classification for $KDDTrain^{+}$ and $KDDTest^{+}$ datasets are 0.45min and 0.07min respectively.

7.4.7 Results on TUIDS Intrusion Dataset

The proposed method was evaluated using our real life intrusion dataset TUIDS. Normal and a specific attack data are used as input to outlier detection method. The identified outlier objects are considered as the specific attack. The results using the measure DR and FPR are given in *Table 7.10*. The average execution time of classification for *Packet Level* and *Flow Level* datasets are 0.44min and 0.19min respectively.

Table 7.10: Experimental Results of TUIDS datasets

Dataset	*DR*			*FPR*
	DoS	*Probe*	*All Attacks*	
Packet Level	76.33	52.16	64.25	1.23
Flow Level	82.21	49.34	65.61	1.62

Table 7.11: Experimental Results of TUIDS PortscanTrain dataset

DR					*FPR*
SYN	*ACK*	*FIN*	*xmas*	*All Attacks*	
81.25	48.84	68.96	70.95	67.50	1.81

7.4.8 Results on TUIDS Portscan Dataset

The proposed method was evaluated using our real life TUIDS *PortscanTrain* dataset. Normal and a specific attack data are used as input to outlier detection method. The identified outlier objects are considered as the specific attack. The results using the measure *DR* and *FPR* are given in Table 7.11. The average execution time of classification for *PortscanTrain* is 0.15min.

7.4.9 Comparison of Results

The summarized results over the KDD Cup 1999 intrusion datasets are made for detection rate (*DR*) on intrusive records and *FPR* over *normal* records for the datasets. Performance evaluation results are compared with Elkan's [215] result and Zhiyi et. al.[216]. The comparison results for the datasets and for its preprocessed one are presented in *Table 7.12*. In performance comparison, our method outperforms other counter parts in terms of *DR* for *R2L* and *U2R* attack categories.

A comparative result of our method and other two methods *LOF*[213] and *ORCA*[214] using three real life private datasets of TUIDS :*Packet level*, *Flow level* and *Portscan* are given in Table 7.13 and 7.14. The performance of our

Table 7.12: Comparative results over *Corrected* KDD Cup dataset

	Elkan's	Zhiyi's	**Proposed Method**
DR for *DoS*	97.1%	25.1%-65.3%	22.1%-32.1%
DR for *R2L*	8.4%	20.9%-79.8%	**55.1%-89.0%**
DR for *U2R*	13.2%	27.6%-75.3%	**71.3%-84.3%**
DR for *Probe*	83.3%	31.6%-68.7%	29.6%-39.8%
DR for All attack	91.8%	48.8%-70.3%	25.74%-35.96%
FPR	0.5%	0.3%-0.4%	0.9%-1.3%

Table 7.13: Result Comparisons over TUIDS datasets

Datasets	*Methods*	*DR*			*FPR*
		DoS	*Probe*	*All Attacks*	
PacketLevel	***LOF***[213]	72.29	46.54	59.42	2.32
	ORCA[214]	68.90	40.80	54.85	1.92
	Our	76.33	52.16	64.25	1.23
FlowLevel	***LOF***[213]	76.49	45.50	61.00	1.85
	ORCA[214]	72.30	43.60	57.95	2.10
	Our	82.21	49.34	65.61	1.62

method shows that it is better than the other two methods.

7.4.10 Discussion

In this chapter, we present definition for outlier and its detection methods. Apart from these, an efficient outlier detection method is presented based on[207] for multidimensional datasets with an application to network anomaly detection. The proposed method is evaluated with various datasets including UCI ML Repository dataset, popular intrusion datasets, KDD Cup 1999 and NSL-KDD, and real time network intrusion TUIDS datasets. The detection performance of the method is good and better than similar methods.

The outlier method is effective in handling mainly rare category attacks like *R2L* and *U2R*.

Table 7.14: Result comparison over TUIDS PortscanTrain dataset

Methods	*DR*					*FPR*
	SYN	*ACK*	*FIN*	*xmas*	*All Attacks*	
LOF[213]	75.45	44.56	63.56	71.20	63.69	2.35
ORCA[214]	78.35	38.98	66.76	69.90	63.50	1.95
Our	81.25	48.84	68.96	70.95	67.50	1.81

The performance of individual classifiers in network anomaly detection, are not equally good for classification of all categories of attack as well as normal instances. There is a possibility of obtaining good classification accuracy for all categories in a dataset with uses of multiple well performed classifiers in appropriate combination. In the *Chapter 8*, a combination of supervised, unsupervised and outlier based methods to achieve best detection performances for both known and unknown attacks is introduced for network anomaly detection.

CHAPTER 8

Multi-Level Hybrid Method for Anomaly Detection

Contents

This chapter proposes a multi-level hybrid intrusion detection method that uses a combination of supervised, unsupervised and outlier based methods to achieve best detection performances for both known and unknown attacks. The method attempts to derive benefits of each participating classifier toward achievement of a best overall cost efficient classification result. The method was evaluated with two benchmarks and three real life intrusion datasets and performs very well in comparison to other competing methods.

8.1 Introduction

With the enormous growth of network-based computer services and the huge increase in the number of applications running on networked systems, the adoption of appropriate security measures to protect against computer and network intrusions is a crucial issue in a computing environment. Intrusion into or attacks on a computer or network system are activities or attempts to destabilize it by compromising security in confidentiality, availability or integrity of the system. As defined in[217], an intrusion detection system (IDS) monitors the events occurring in a computer system or a network and analyzes them for signs of intrusions. Network-based intrusion detection is generally implemented using two approaches: rule-based[20] and anomaly-based This chapter focusses on anomaly based intrusion detection using a hybrid method that combines supervised, unsupervised and outlier approaches.

Before initiating a detailed discussion on hybrid methods for network anomaly detection, let us introduce supervised, unsupervised and outlier based detection methods in brief to provide the necessary background.

8.1.1 Supervised Intrusion Detection

Based on the machine learning method used, anomaly detection can be classified into two different categories: supervised[70] and unsupervised. In supervised anomaly detection, normal behaviour models of systems or networks are established by training with labeled or purely normal datasets. These normal behaviour models are used to classify new network connections. The system generates an alert if a connection is classified as malign or abnormal behaviour. For example, ADAM[22] is a supervised anomaly-based as well as misuse-based NIDS. There is an SVM-based IDS[179] that combines a hierarchical clustering algorithm, a simple feature selection procedure, and the SVM[167,168] technique. The hierarchical clustering algorithm provides the SVM with fewer, abstracted, and highly qualified training instances that are derived from the KDD Cup 1999 training set. For feature selection, it removes one feature from the original dataset, redoes the experiment, then compares the new results with the original results and determines the importance of the feature by the change in false positives or accuracy rates. The SVM model is used to classify network traffic data using the selected features. However, in practice, to train a supervised anomaly-based approach, labeled or purely normal data is not easily available. It is time consuming to acquire and error-prone to manually classify and label instances as benign or malign. In supervised anomaly-based detection, obtaining labeled or purely normal data is a critical issue. A set of training as well as testing real life labeled intrusion datasets TUIDS[92] were generated, as discussed in *Chapter* 3

8.1.1.1 Discussion

Based on our experimental study of various existing well-known supervised classifiers using benchmark and real life intrusion data, we observe the following.

- Tree based classifiers (like C4.5 and Random forests) can perform consistently well with a majority of datasets, but they are unstable in nature[218,219]. Slight change in the threshold values may cause significant change in performance. In addition, appropriate selection of these parameters is a difficult task.

- Clustering based supervised methods such as IBk[220] or kNN show stable performance, least affected by changes in the threshold values. However, their performance is not guaranteed for all datasets and for all classes[221].

- Probabilistic supervised classifiers (like Naive Bayes or Bayesian Clustering) are good but do not perform expectedly for all classes[161,24]

- Supervised classifiers are dependent on labeled training data.
- A Major issues with supervised classifiers is the lack of pure and sufficient normal training traffic.
- Supervised methods are incapable of identifying unknown attacks.

8.1.2 Unsupervised Intrusion Detection

Unsupervised anomaly detection approaches work without any training data. In other words, these models are used on unlabeled.or unclassified data when they attempt to find intrusions lurking inside the data. A number of IDSs employ unsupervised anomaly-based approaches [197,66]. The biggest advantage of the anomaly detection approach is the detection of unknown intrusions without any previous knowledge of intrusions. However, this approach fails to detect all intrusions. In addition, the false detection rate tends to be higher if the behaviour of some of the intrusions is not significantly different from the considered normal behaviour. model. In network-based intrusion detection, the threat arises from new or previously unknown intrusions. The prefered detection approach for novel intrusions is anomaly-based instead of rule-based. Unsupervised anomaly-based detection can address this issue of novel intrusion detection without prior knowledge of intrusions or using purely normal data.

In literature [66,65,67,68], clustering and classification are widely used methods for anomaly-based supervised or unsupervised detection of intrusions. Clustering is a method of grouping objects based on similarity among them. The similarity within a cluster is high and the dissimilarity among clusters is high as well. Clustering is a kind of unsupervised method of analysis [69]. This method is performed on unlabeled data. It assigns similar data to the same class and dissimilar data to different classes. Unsupervised anomaly-based detection often tries to cluster the test dataset into groups of similar instances which may be either intrusion or normal data. Although using a clustering method in unsupervised anomaly-based detection for intrusion generates many clusters, the stability of these clusters and their labeling are still difficult issues faced by this approach. In order to label clusters, an unsupervised anomaly-based detection approach models normal behaviour using two assumptions [70]: (i) The number of normal instances vastly outnumbers the number of anomalies and (ii) Anomalies themselves are qualitatively different from normal instances. If these assumptions hold, intrusions can be detected based on cluster size. Larger clusters correspond to normal data, and smaller clusters correspond to intrusions. But this method is likely to produce high false detection rate as the assumptions are not always true in practice. For example, in the denial of service category of intrusion, a large number of very similar instances are generated resulting in larger clusters than the cluster corresponding to normal behaviour..On the other hand in remote to local (R2L) and user to root (U2R) categories of intrusion, legitimate and illegitimate users are difficult to distinguish and their numbers of occurrence instances may not be significantly larger. These intrusions may get included in the normal behaviour model. Consequently, these can raise the false detection rate.

8.1.2.1 Discussion

Detecting new attacks is a challenging task for intrusion detection methods. Anomaly based intrusion detection is a possible method for novel attack detection. Supervised classification methods are dependent on labeled training data. On the other hand, unsupervised anomaly based intrusion detection does not require labeled training data for detection. The unsupervised methods are usually examples of the clustering approach. Here the attack detection is based on certain assumptions about the data. Therefore, the false positive rate of detection is generally high.

8.1.3 Outlier Based Intrusion Detection

Outliers refer to data points that are very different from the rest of the data based on appropriate measures. Such data points often contain useful information regarding unusual behaviour of a system described by the data. These anomalous data points are usually called outliers. Outlier detection is widely used in finding anomalous activity in telecommunication, credit card fraud detection, detecting symptoms of new diseases and novel network attack detection The major challenge for outlier detection in intrusion detection is to handle the huge volume of mixed type- numerical and categorical data. So, outlier detection schemes need to be computationally efficient to handle these large sized inputs in fast. An outlier can be an observation that is distinctly different or is at a position of abnormal distance from other values in the dataset. Detection of abnormal behaviour is based, usually, on relevant features extracted from network traces, packet or flow data An intrusion can be detected by finding an outlier whose features are distinctly different from the rest of the data. Outliers can often be individuals or groups of objects exhibiting behaviour outside the range of what is considered *normal*. Outlier detection in finding anomalies is beneficial in case of rare intrusion instances. For example, in the KDD Cup 1999 datasets, *R2L* and *U2R* categories are rare in comparison with categories of *DoS* or *Probe*. In the usual methods of classification, these rare category instances may remain undetected. For rare attack detection, outlier based methods are very effective. Outlier based intrusion detection methods such as LOF[114], LOADED[113], Jin et. al:[207] or ODMAD[79] focus on mining outliers by using density distribution of the objects, or finding the distance vectors among the objects, or by finding anomaly scores of objects. Depending on the approach used in outlier detection, the methodologies can be broadly classified[61] as distance-based density-based and machine learning or soft computing based.

8.1.3.1 Discussion

Outlier detection searches for objects that do not obey rules and expectations valid for the major part of the data. Outlier detection methods are mainly important for identification of attacks that are rare in a large intrusion dataset. Outlier based methods are used as supervised anomaly intrusion detection methods.

8.1.4 Multi-level IDS

Several multi-level hybrid IDSs have been proposed recently to deal with the complexity of the intrusion detection problem by combining different machine learning techniques. Pan et al.[222] combined neural networks and the C4.5 decision tree algorithm to detect a variety of attacks. Depren et al.[152] used a hybrid IDS consisting of an anomaly detection module, a misuse detection module and a decision support system. Zhang et al.[46] combined misuse detection and anomaly detection components using the random forests algorithm.

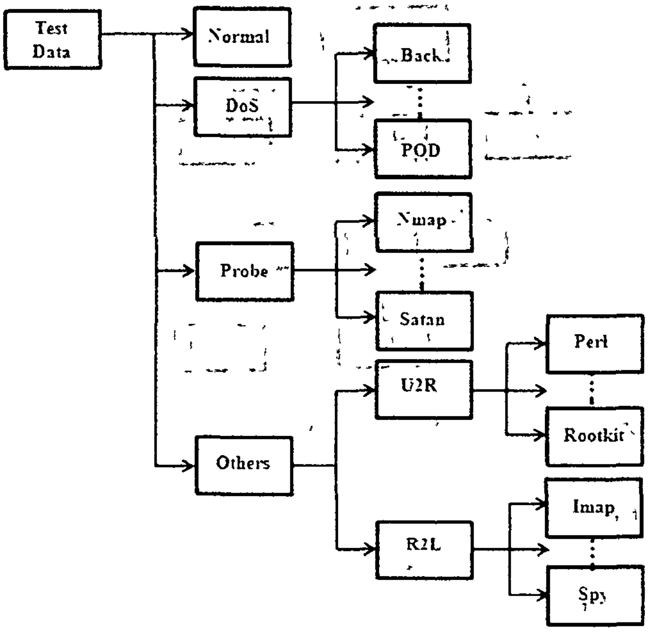


Figure 8.1: Multiple Level Tree Classifier[1]

Hwang et al.[153] proposed a hybrid system combining signature-based intrusion detection and anomaly detection to detect novel unknown attacks. In order to further increase the intrusion detection rate, as well as to simplify the algorithm, Xiang et al.[1] proposed a multiple-level tree classifier to design an IDS containing three levels of decision tree classification.

A multiple-level tree classifier[1] as given in Figure 8.1 uses the KDD CUP 99 dataset for training as well as testing. The first round of classification categorizes the test data into their predicted attack type (DoS, Probe, Others or Normal). The second round splits Others into U2R and R2L groups. The third round further classifies the attacks into their specific attacks (e.g., back, land, etc.). The method uses the C4.5 algorithm to construct decision trees for classification. The information gain measure is used to select the test attribute at each node in the tree. The attribute with the highest information gain is chosen as the test attribute for the current node. This attribute minimizes the information needed to classify samples in the resulting partitions.

A three-level hybrid intrusion detection system reported in[2], utilizes decision trees, Naïve Bayes and Baysian Clustering as shown in Figure 8.2. The

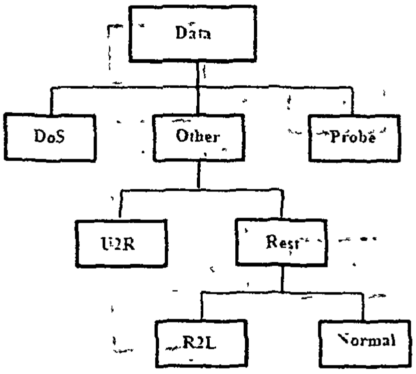


Figure 8.2: Architecture of the Three Level Hybrid Classifier[2]

training and test data used in this work is the KDD CUP 1999 dataset. In stage 1 of classification, it uses the C4.5[157] decision tree learner to classify the dataset into three categories: DoS, Probe and other. In stage 2, the U2R and Rest records are separated out with Naïve Bayes[124] algorithm. The last stage uses Bayesian clustering[223] to split Rest into Normal and R2L categories.

The Multiple-level Hybrid Classifier[3] combines supervised tree classifiers and unsupervised Bayesian clustering[223] to detect intrusions. The KDD CUP 99 dataset is used to evaluate the method. The method consists of four stages of classification as given in Figure 8.3 In stage 1 of classification, it categorizes the test data into three categories (DoS, Probe, Others). U2R and R2L, and the Normal connections are classified as Others in this stage. The second stage splits Others into Attack and Normal categories The third stage separates the Attack class from stage 2 into U2R and R2L. The fourth stage further classifies the attacks into more specific attack types.

In[4], the Multilevel Network Security Structure (MNSS) is presented as shown in Figure 8.4. It integrates firewall, intrusion detection and reporting into one structure The intrusion detection system sets three sensors on the MNSS structure. They are set in DMZ, in front of firewall (LAN) and behind firewall (WAN). The function of the sensor in front of the firewall (LAN) is mainly to check whether someone attacks the network or not. The function of the sensor behind the firewall (WAN) is to monitor attacks in the wide-area network. Finally, the sensor in the DMZ is to ensure servers are safe. This structure ensures that it cannot lose any probable attack and also know where

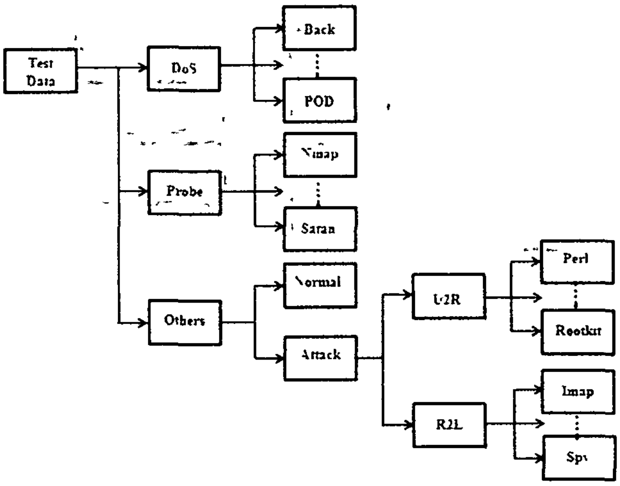


Figure 8.3: Structure of the Multiple Level Hybrid Classifier[3]

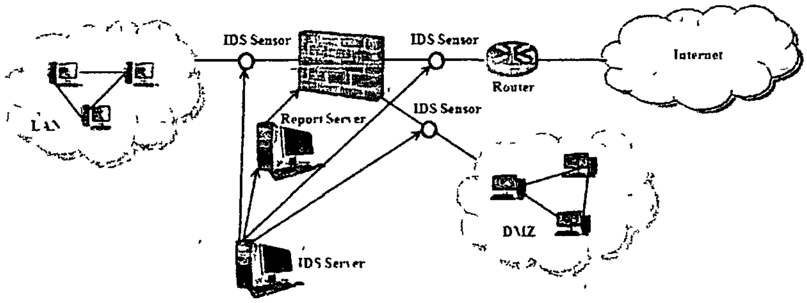


Figure 8.4: Multilevel Network Security Structure[4]

attack happened through the structure.

Youssif et al.[5] propose a multi-level intrusion detection system, ML-IDS. It is capable of detecting any type of network attacks. The architectuie of ML-IDS is given in Figure 8 5 The current implementation of ML-IDS uses two types of attack detection techniques: flow based and protocol based. The flow-based approach employs a set of rules that, if violated, flags the traffic flow as being anomalous. The protocol behaviour approach views network protocol operations in terms of a finite state machine, and flags illegal sequences of state transitions as being anomalous A fusion module that implements the least squares algorithm is used to combine the decision produced by each type of anomaly analysis and thus significantly reduce the attack detection

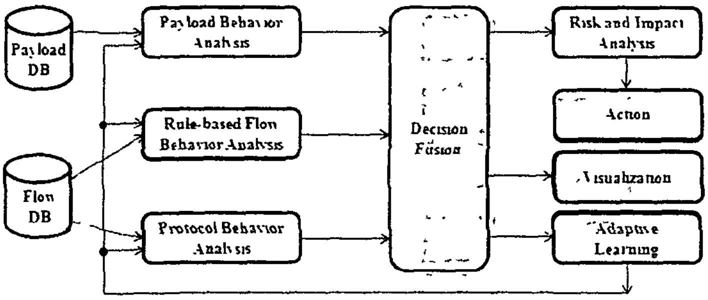


Figure 8.5: Architecture of multi-level IDS[5]

error. An autonomic control and management environment is used to automate the performance, fault and security management of network centric systems. The architecture of ML-IDS consists of modules for online monitoring and filtering, multi-level behaviour analysis, decision fusion, risk and impact analysis, action, visualization, and adaptive learning. The online monitoring modules use existing software tools to monitor network and host traffic. In multi-level behaviour analysis, rule-based behaviour, protocol behaviour and payload analysis are used to detect anomalous events in network operations. If anomalous behaviour is detected by any of the three systems, an alert is sent to the decision fusion module. The decision fusion module fuses the decisions produced by each type of behaviour analysis modules using the least squares linear regression algorithm. When the risk and impact analysis module receives one or more alerts from the behaviour analysis module, it determines what part of the system will be affected by the detected attack using its knowledge repository. Once the action module receives the results from the risk and impact analysis, it determines the appropriate automatic actions to be executed using its knowledge repository. The visualization module is used to give administrators a better understanding of the activities taking place in the network and also allows them to interact with resources. The adaptive learning module will add the mis-detected patterns and expand the normal and the abnormal profiles repository. Thus, the supervised learning module is activated to generate new rules, or to expand the trained profiles so as to capture new types of anomalous behaviours.

8.1.5 Discussion

In multi-level intrusion detection, several methods of intrusion detection participate to maximize their positive achievements and to minimize their weak ones to achieve high detection rate and minimum the false positive rate. In Table 8.1, we compare several existing multi-level NIDSs based on parameters such as detection, learning approach (supervised, unsupervised or both), number of levels of classification, method (misuse, anomaly or both), time of detection (realtime or non-realtime), nature of processing (centralized or distributed), data gathering mechanism (centralized or distributed), dataset used, attribute space (full or reduced), unknown attack handling and the algorithmic approach. Based on our experimental study of various existing well known multi-level IDSs using benchmark intrusion datasets, we observe the following.

- Using a multi-level hybrid classifier is a potential approach for network anomaly detection to obtain a low false positive rate (FPR).
- Such a method can be tuned to enable detection of unknown attacks as well as unknown normal traffic.
- The number of levels in a MLH-IDS (multi-level hybrid IDS) is a key issue. More the number of levels, the more will be the computational overhead, in a strictly sequential system.
- For the supervised component, the creation of an appropriate training sample is a major issue.
- For the unsupervised component, the creation of clusters is an important issue.

Developing an MLH-IDS with (i) a minimum number of levels, (ii) a supervised classifier powered by appropriate training samples, (iii) an unsupervised classifier supported by necessary stability analysis, is of utmost importance.

8.2 Background

In clustering, an analysis of the stability of the obtained clusters is necessary. Stability analysis for cluster validity uses indexes viz., Dunn index[72],

Table 8.1: Existing NIDSs using Multi-level Approach

Author	*Year of Publish*	*No. of Levels*	*Learning Approach*	*Detection Method*	*Detection Time*	*Data Processing*	*Data Gathering*	*Dataset Used*	*Attribute Spaces*	*Unknown Attack Handling*	*Algorithmic Approach*
Xiang et al [1]	2004	3	B	B	NR	C	C	K	Rd	N	C4 5
Depren et al [152]	2005	2	B	B	NR	C	C	K	F	N	Misuse & Decision support
Chen et at [4]	2005	3	S	B	NR	C	C	K	F	N	Signature based
Zhang et al [46]	2006	2	U	B	NR	C	C	K	Rd	Y	Random Forest
Hwang et al [153]	2007	2	B	B	NR	C	C	K	F	Y	Misuse & Anomaly based
Xiang et al [3]	2008	4	B	B	NR	C	C	K	Rd	Y	Bayesian Clustering
Youssif et al [5]	2008	3	B	B	R	C	C	K	F	Y	Rule based & Protocol analysis
Hui Lu et al.[2]	2009	3	U	A	NR	C	C	K	F	N	Bayesian Clustering

Note-*S-Supervised/U-Unsupervised/B-Both, M-Misuse/A-Anomaly/B-Both, K-KDD Cup
R-Realtime/NR-Non realtime, C-Centralised/D-Distributed, Rd-Reduced/F-Full, Y-Yes/N-No

C-index[224], Davies Bouldin's index (DB)[74], Silhouette index (S)[75] and Xie-Beni index (XB)[76].

Dunn index is computed to find compact and well separated clusters. It depends on d_{min} and d_{max}. d_{min} denotes the smallest distance between two objects from different clusters and d_{max} is the largest distance between two elements inside a cluster. Dunn index[72] is given by Equation (8.1), where Dunn $\in [0, \infty]$, larger values of Dunn indicating better clustering.

$$Dunn = \frac{d_{min}}{d_{max}}. \tag{8.1}$$

C-index is used to find cluster quality when the clusters are of similar sizes. C-index[224] is defined as

$$C = \frac{S - S_{min}}{S_{max} - S_{min}} \tag{8.2}$$

where S is the sum of distances over all pairs of objects from the same cluster, n is the number of such pairs S_{min} and S_{max} are the sums of the n smallest distances and n largest distances, respectively C $\in [0, 1]$, and the smaller the values of C, the better are the clusters.

Davies Bouldin's index is the ratio of the sum of within-cluster to between-cluster separation. Davies Bouldin's index[74] is given by

$$DB = \frac{1}{n} \sum_{i=1, i \neq j}^{n} max \left(\frac{\delta_i + \delta_j}{d(c_i, c_j)} \right). \tag{8.3}$$

In Equation (8.3), n is the number of clusters; δ_i is the average distance of all patterns in the i^{th} cluster to their cluster center, c_i; δ_j is the average distance of all patterns in cluster j to their cluster center, c_j; and $d(c_i, c_j)$ represents the proximity measure between the cluster centers c_i and c_j. Lower value of DB indicates better clusters.

Silhouette index[75] is computed for a cluster to identify tightly separated groups. It is defined as

$$SI = \frac{b_i - a_i}{max\{a_i, b_i\}} \tag{8.4}$$

where a_i is the average dissimilarity of the i^{th} object to all other objects in the same cluster; b_i is the minimum of average dissimilarity of the i^{th} object to all objects in other clusters.

Xie-Beni index[76] is given by Equation (8.5). A function calculates the compactness π of data in the same cluster and a second computes the separateness s of data in different clusters. $\pi = \frac{\delta_i}{n_i}$, is the compactness of i^{th} cluster where δ_i is the variation, i.e., summation of the squares of deviation of the data points in the cluster and n_i is the number of points in the cluster. $s = (d_{min})^2$ is the separation and $d_{min} = min\,||k_i - k_j||$, where d_{min} is the minimum distance between cluster centers k_i and k_j.

$$XB = \frac{\pi}{s} \tag{8.5}$$

Smaller values of π means that the clusters are more compact and larger values of s mean the clusters are well separated. Thus a smaller XB reflects that the clusters have greater separation from each other and are more compact

There are several other stability indices found in the literature. We have reported here a selected few only. These distance based indices help judge the quality of clustering based on compactness. Each index has its own advantages and disadvantages. So, it is necessary to suppress the individual biases, while deciding the stability of clustering results depending on these indices.

8.3 Problem Formulation

For a given set of classifiers $C_1, C_2, \dots, C_N$ and for a given set of class specific sample datasets $S_1, S_2, \dots, S_M$, the objective of a multi-level hybrid classifier is to predict the most appropriate *class ID* for a given test instance $t_i \in D$, the test dataset, by appropriate use of the classifier for given classes at a particular level with reference to the training sample(s) S_i's.

8.4 Contributions

We design an effective MLH intrusion detection method that can leverage the merits of the supervised, unsupervised and outlier detection methods (reported in the preceding *Chapters* 5, 6 and 7). The following are the contributions of this chapter.

- A hybrid multi-level classifier based on supervised, unsupervised and outlier detection algorithms.
- A cluster stability analysis and a robust cluster labeling technique.
- Performance evaluation of the hybrid classifiers in terms of several real life and benchmark datasets.

The problem of detecting new attacks poses a special challenge to the research community. Within such a context, intrusion detection systems using the multi-level classifiers have produced remarkable improvements. In this chapter we present a multi-level hybrid intrusion detection system (MLH-IDS), which is capable of detecting network attacks. The implementation of MLH-IDS uses three levels of attack detection techniques for classification: supervised, unsupervised and outlier based. We discuss in further details of our method to implement each module and evaluate performance against a wide range of network attacks.

8.4.1 Supervised Classification Method

The proposed classification technique uses a training algorithm to create a set of representative clusters from the available labeled training objects. Unlabeled test objects are then inserted in these representative clusters based on similarity calculations and thus get labels of the clusters in which they are inserted. The clustering algorithm (*CatSub*) is an application of the algorithm presented in[225]. Details of this method are reported in *Chapter* 5.

8.4.2 Unsupervised Classification Method

The unsupervised classification method uses the k-point algorithm[226] to create a set of representative clusters from the available unlabeled objects in the data. Initially, the method considers k objects randomly from the dataset. The data objects are then gathered around these selected points (or objects) based on various attribute similarities The clusters are formed using two similarity measures: (i) similarity between two objects, and (ii) similarity between a cluster and an object. The method produces various clusters. Details of this method are reported in *Chapter* 6.

8.4.3 Outlier based Classification Method

We have developed an outlier mining method based on symmetric neighborhood relationships[207]. For each object of dataset, a forward neighbor outlier factor is estimated by finding its nearest neighbors set and the forward nearest neighbors set of the data objects to identify outliers. Details of this outlier finding method are presented in *Chapter* 7.

8.5 Proposed MLH-IDS Framework

8.5.1 Proposed Architecture

The selection of supervised, unsupervised or outlier based classifier at a particular level for a given dataset is based on the desired classification accuracy of the individual classifier for a given dataset. The overall structure of the multi-level hybrid classifier we have developed is shown in Figure 8.6.

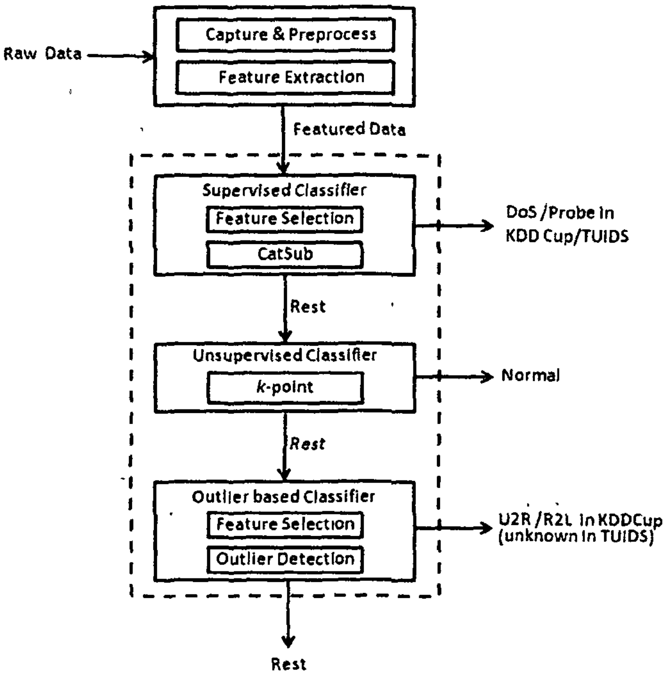


Figure 8.6: Architecture of Multi-level Hybrid Classifier

A model with three stages of classification is used for the hybrid classifier. The first level of classification categorizes the test data into three categories *DoS*, *Probe* and Rest (unclassified). *U2R*, and *R2L*, and the *Normal* connections are classified as Rest at this stage. The second level splits Rest into Normal and Rest categories. The third level separates Rest into *U2R*, *R2L* and Rest. The main purpose at level 1 is to extract as many *U2R*, *R2L* and *Normal* connections as possible accurately from the data using a supervised classifier model. This is because *U2R* and *R2L* are generally quite similar to normal connections. *DoS* and *Probe* attack connections on the other hand are generally more different. At level 2, the Rest category is classified as Normal and Rest (i.e., class containing attacks) using an unsupervised classifier model The *U2R* and *R2L* categories are extracted from Rest using the outlier based classifier model and the remaining elements are classified as Rest (unknown)

In level 1 classification, different attack records of only *DoS* and *Probe* are taken for training as given in *Chapter* 5. Profiles are created for *DoS* and *Probe* categories in training. The profiles are used for testing the whole dataset. The records are classified into three categories, *DoS*, *Probe* and Rest. The Rest records are forwarded for level 2 classification. In level 2 classification, records are classified using the unsupervised classifier algorithm called *k-point* as described in *Chapter* 6. The records are classified into two categories, *Normal* and Rest. The Rest category records are finally forwarded to level 3 classification. In level 3 classification, an outlier based classification method is used, as described in *Chapter* 7. In high dimensional data, feature selection using mutual information (MI) is a widely accepted approach. In this level, MI[84] based relevant features of *U2R* and *R2L* are used in the model to classify the *U2R* and *R2L* categories. Other than *U2R* and *R2L* categories, the remaining objects are classified as Rest (unknown).

8.5.2 Cluster Labeling Technique

We analyze the stability of clusters obtained from the k-point algorithm. We propose an ensemble based stability analysis technique based on Dunn index [72], C-index[224], Davies Bouldin's index (DB)[74], Silhouette index (S)[75] and Xie-Beni index (XB)[76] for cluster validity (shown in Figure 8.7).

Table 8.2: Cluster Stability Criteria

Stability measure	*Range of value*	*Criteria for better cluster*
Dunn index	$(0,\infty)$	Maximized
C-index	(0,1)	Minimized
Davies Bouldin's index	$(0,\infty)$	Minimized
Silhouette index	(-1,1)	Near 1
Xie-Beni index	(0,1)	Minimized

The criteria used for labeling the cluster stability analysis are given in *Table 8.2*. Cluster results are passed to a function to compute all the indices for each of the clusters C_1, C_2, ..., C_k, and if the value found is within the acceptable range, then the cluster index, CI_i is stored as 1, otherwise it is

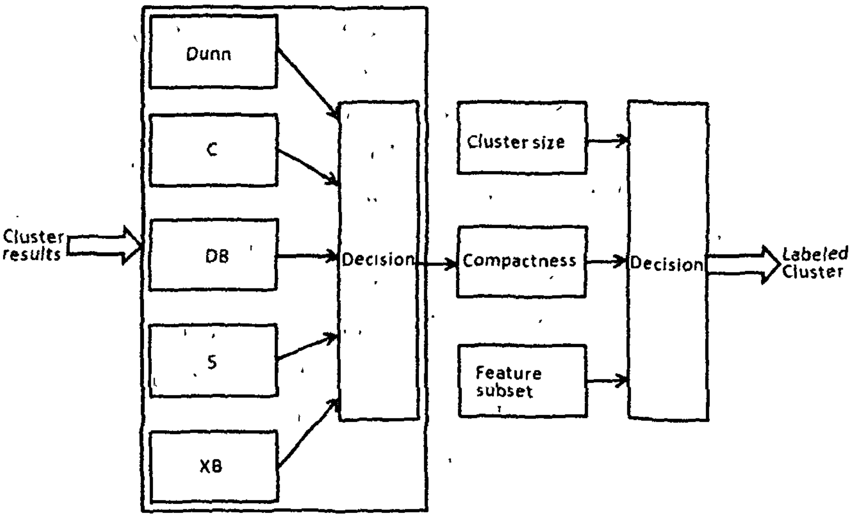


Figure 8.7: Architecture of multiple indices cluster labeling

assigned 0, as defined below

$$CI_i = \begin{cases} 1, & j^{th} \text{ index value, } I_j \text{ is within its valid range} \\ 0, & \text{otherwise} \end{cases} \tag{8.6}$$

Finally, we consider the number of occurrences of 1 to decide the compactness of a cluster. This becomes input to the subsequent labeling task along *with two the other measures, viz., cluster size and dominant feature subset* shown in Fig. 8.7. To compute the dominating feature subset, we use rough sets and a modified genetic algorithm based method described in[227].

8.5.3 Complexity analysis

MLH-IDS is comprised of three individual methods. The time complexities of the methods are -(i) $O(ncd) + O(c)$ for *CatSub* (ii) $O(ncd)$ for k-point and (iii) $O(n^2)$ for the outlier based method. The time complexity of MLH-IDS is the sum of these three methods.

Table 8.3: Attack distribution in benchmark intrusion datasets

KDD Cup 1999				NSL-KDD			
Datasets	*Normal*	*Attack*	*Total*	*Datasets*	*Normal*	*Attack*	*Total*
Corrected KDD	60593	250436	311029	*KDDTest*+	9710	12834	22544
10-percent KDD	97278	396743	494021	*KDDTrain*+	67343	58630	125973

Table 8.4: Attack distribution in TUIDS intrusion datasets

Packet Level				Flow Level			
Datasets	*Normal*	*Attack*	*Total*	*Datasets*	*Normal*	*Attack*	*Total*
Training	71785	50142	121927	*Training*	23120	29723	52843
Testing	47895	38370	86265	*Testing*	16770	23955	40725

8.6 Experimental Results

To evaluate the performance of our method, all experiments were carried out on an Intel workstation with Intel core 2 Quad @2.4GHz, 2 GB RAM, 160GB HDD.

8.6.1 Dataset Used

The algorithm was tested on two benchmark intrusion datasets, viz., KDD Cup 1999[81] and NSL-KDD[82] datasets. The algorithm was also evaluated with three real life TUIDS[92] intrusion datasets. The description of the datasets are given in *Chapter* 3. The Tables 8.3, 8.4 and 8.5 show the attack distributions of the datasets.

8.6.2 Benchmark Intrusion Dataset

8.6.2.1 Results using KDD Cup 1999 Dataset

The confusion matrices for KDD Cup 1999 dataset on the *Corrected KDD* and the 10 *percent KDD* are shown in Table 8.6 and Table 8.7, respectively. The values of the validity measure *Recall* are 0.9999 and 0.9875 for *DoS* and

Table 8.5: Attack distribution in TUIDS intrusion datasets (continue)

Portscan			
Datasets	*Normal*	*Attack*	*Total*
PortscanTrain	2445	39215	41660
PortscanTest	1300	28615	29915

Table.8.6: Confusion matrix for *Corrected KDD* dataset

Actual Class	Predicted Class							
	DoS	*Probe*	*Normal*	*U2R*	*R2L*	Sum	Recall	1-Prc*
DoS	229828	2	23	0	0	229853	0.9999	0.0001
Probe	7	4114	42	0	3	4166	0.9875	0.0044
Normal	6	15	57605	1	2966	60593	0.9507	0.0188
U2R	0	0	10	57	3	70	0.8143	0.0272
R2L	0	1	1027	2	15317	16347	0.9370	0.1625
Sum	229841	4132	58707	60	18289	311029		

Re-substitution error rate=0.0132 PCC=98.68%

Note- *1-Precision

Probe, respectively in Table 8.6 and 1 0000, 0.9963 and 0.9994 for *DoS*, *Probe* and *Normal*, respectively in Table 8.7, indicating good performance of our technique The results also indicate satisfactory classification rates for *R2L* and *U2R* categories of attack. Smaller values of 1-*Precision* indicate low false positives. 1-*Precision* values for *DoS* and *Probe* are 0.0001 and 0.0044, respectively in Table 8.6 and *DoS*, *Probe* and *Normal* are 0.0000, 0.0104 and 0.0012, respectively in Table 8.7. Average execution time for classification is 1.36 minute and 1.91 minute for *Corrected KDD* and the 10 *percent KDD*, respectively.

In cross evaluation 10-*percentKDD* dataset is used for training and *Corrected KDD* dataset is used for testing. The attack records which are present in *Corrected KDD* dataset but do not exist in 10-*percentKDD* dataset, are marked as *unknown*. The result is given in Table 8.8. In this case, the majority of *unknown* attacks is misclassified as *R2L* category and *PCC* is found as 95.85% with error rate 0.0415.

8.6.2.2 Results on NSL-KDD Dataset

The confusion matrices for NSL-KDD dataset on the $KDDTrain^+$ and the $KDDTest^+$ are shown in Table 8.9 and Table 8.10, respectively. The performance results are similar to the results using the KDD Cup 1999 dataset. The values of the validity measure *Recall* are 0.9994, 0.9946 and 0.9948 for *DoS*, *Probe* and *Normal*, respectively in Table 8.9 and 0.9980, 0.9905 and

Table 8.7: Confusion matrix for 10-*percent KDD* dataset

Actual	Predicted Class							
Class	*DoS*	*Probe*	*Normal*	*U2R*	*R2L*	Sum	Recall	1-Prc*
DoS	391450	1	7	0	0	391458	1.0000	0.0000
Probe	0	4092	15	0	0	4107	0.9963	0.0104
Normal	1	42	97221	0	14	97278	0.9994	0.0012
U2R	0	0	6	44	2	52	0.8462	0.0435
R2L	0	0	53	2	1071	1126	0.9512	0.0147
Sum	391451	4135	97302	46	1087	494021		
Re-substitution error rate=0.0003					PCC=99.97%			

Note- *1-Precision

Table 8.8: Confusion matrix for cross evaluation with 10-*percentKDD* for training and *Corrected KDD* for testing

Actual	Predicted Class								
Class	*DoS*	*Probe*	*Normal*	*U2R*	*R2L*	*Ukn*@	Sum	Recall	1-Prc*
DoS	221903	102	702	19	42	530	223298	0.9938	0.0047
Probe	43	2247	22	4	10	51	2377	0.9453	0.2439
Normal	433	192	58563	14	441	950	60593	0.9665	0.0155
U2R	0	0	1	35	1	2	39	0.8974	0.6316
R2L	3	182	170	5	5273	360	5993	0.8798	0.2620
Ukn@	565	249	26	18	1378	16493	18729	0.8806	0.1030
Sum	222947	2972	59484	95	7145	18386	311029		
Re-substitution Error=0.0210						PCC=97.90%			

Note- *1-Precision @Unknown

Table 8.9: Confusion matrix for $KDDTrain^+$ dataset

Actual Class	Predicted Class							
	DoS	*Probe*	*Normal*	*U2R*	*R2L*	Sum	Recall	1-Prc*
DoS	45900	4	23	0	0	45927	0.9994	0.0025
Probe	0	11593	63	0	0	11656	0.9946	0.0080
Normal	114	89	66993	5	142	67343	0.9948	0.0019
U2R	0	0	9	43	0	52	0.8269	0.1224
R2L	0	0	31	1	963	995	0.9678	0 1285
Sum	46014	11686	67119	49	1105	125973		
Re-substitution error rate=0.0038				PCC=99.62%				

Note- *1-Precision

0.9868 for *DoS*, *Probe* and *Normal*, respectively in Table 8.10. The results also show satisfactory *Recall* values for *R2L* and *U2R*. 1-*Precision* values for *DoS* and *Probe* are 0.0025 and 0.0080, respectively in Table 8.9 and are 0.0009 and 0.0099, respectively in Table 8.10. The average execution time of classification is 0.67 minute and 0.11 minute for $KDDTrain^+$ and $KDDTest^+$ datasets, respectively.

For cross evaluation, the $KDDTrain^+$ dataset is used for training and the $KDDTest^+$ dataset is used for testing. The attack records which exist in $KDDTest^+$ dataset but do not exist in $KDDTrain^+$ dataset, are marked as *unknown*. The result is given in Table 8.11. In this case the majority of *unknown* attacks is misclassified as *DoS*, *Probe* and *R2L* categories and *PCC* is found as 95.68% with error rate 0.0432.

8.6.3 Real-life Intrusion Dataset

8.6.3.1 Results on TUIDS Packet Level Dataset

The confusion matrix for all-attacks categories on the *Packet Level* dataset is shown in Table 8.12. *Recall* values for *DoS*, *Probe* and *Normal* are respectively 0.9989, 0.9795 and 0.9973 as given in Table 8.12. 1-*Precision* values for *DoS*, *Probe* and *Normal* are respectively 0.0015, 0.0213 and 0.0024 in Table 8.12. The results indicate that the detection rate of our method is high with low false positives. The average execution time of classification is 0.64

Table 8.10: Confusion matrix for $KDDTest^+$ dataset

Actual Class	Predicted Class							
	DoS	*Probe*	*Normal*	*U2R*	*R2L*	Sum	Recall	1-Prc*
DoS	7443	0	14	0	1	7458	0.9980	0 0009
Probe	2	2399	16	0	5	2422	0.9905	0.0099
Normal	5	24	9582	0	99	9710	0.9868	0.0124
U2R	0	0	3	57	7	67	0.8507	0.0500
R2L	0	0	87	3	2797	2887	0.9688	0.0385
Sum	7450	2423	9702	60	2909	22544		
Re-substitution error rate=0.0118				PCC=98.82%				

Note- *1-Precision

Table 8.11: Confusion matrix for cross evaluation with $KDDTrain^+$ dataset for training and $KDDTest^+$ dataset for testing

Actual Class	Predicted Class								
	DoS	*Probe*	*Normal*	*U2R*	*R2L*	*Ukn*@	Sum	Recall	1-Prc*
DoS	5680	0	14	0	1	46	5741	0.9894	0.0363
Probe	2	1072	16	0	5	11	1106	0 9693	0.1104
Normal	5	24	9515	0	89	77	9710	0.9799	0.0156
U2R	0	0	3	31	2	1	37	0.8378	0.0723
R2L	0	0	79	3	1913	179	2174	0 8799	0.0385
Ukn@	207	109	39	9	52	3360	3776	0.8898	0.0855
Sum	5894	1205	9666	43	2062	3674	22544		
Re-substitution error rate=0.0432						PCC=95.68%			

Note- *1-Precision @Unknown

Table 8.12: Confusion matrix for *Training* data in *Packet Level* dataset

Actual Class	Predicted Class					
	DoS	*Probe*	*Normal*	Sum	Recall	1-Prc*
DoS	42545	9	38	42592	0.9989	0 0015
Probe	20	7395	135	7550	0.9795	0.0213
Normal	42	152	71591	71785	0.9973	0.0024
Sum	42607	7556	71764	121927		
Re-substitution error rate=0.0032				PCC=99.68%		

Note- *1-Precision

Table 8.13· Confusion matrix for cross evaluation with *Training* data for training and *Testing* data for testing in *Packet Level* dataset

Actual Class	Predicted Class						
	DoS	*Probe*	*Normal*	*Ukn@*	Sum	Recall	1-Prc*
DoS	24494	95	289	205	25083	0.9765	0.0500
Probe	59	6374	535	789	7757	0.8217	0.0542
Normal	634	265	45624	1372	47895	0 9526	0.0586
Ukn@	595	5	15	4915	5530	0.8888	0.3250
Sum	25782	6739	48463	7281	86265		
Re-substitution error rate=0.0563					PCC=94.37%		

Note- *1-Precision @Unknown

minute.

For cross evaluation of the *Packet Level* dataset, *Training* data is used for training and *Testing* data is used for testing. The attack records which exist in *Testing* but do not exist in *Training* for *Packet Level* dataset are marked as *unknown*. The results are given in Table 8.13. In this case a majority of *unknown* attacks is misclassified as *DoS* and *Normal*, and *PCC* is found as 94.37% with error rate 0.0563.

8.6.3.2 Results on TUIDS Flow Level Dataset

The confusion matrix for all-attacks categories on the *Flow Level* dataset is shown in Table 8.14. *Recall* values for *DoS*, *Probe* and *Normal* are respectively 0.9985, 0.9891 and 0.9864 in Table 8.14. 1-*Precision* values for *DoS*,

Table 8.14: Confusion matrix for *Training* data in *Flow Level* dataset

Actual Class	Predicted Class					
	DoS	*Probe*	*Normal*	Sum	Recall	1-Prc*
DoS	21408	11	22	21441	0.9985	0.0058
Probe	26	8191	65	8282	0.9891	0.0271
Normal	98	217	22805	23120	0.9864	0.0038
Sum	21532	8419	22892	52843		
Re-substitution error rate=0.0083				PCC=99.17%		

Note- *1-Precision

Probe and *Normal* are 0.0058, 0.0271 and 0.0038, respectively in Table 8.14. The result indicates that the detection rate of our method is high with low false positives. The average execution time for classification is 0.25 minute.

For cross evaluation of the *Flow Level* dataset, *Training* data is used for training and *Testing* data is used for testing. The attack records which exist in *Testing* but do not exist in *Training* for the *Flow Level* dataset are marked as *unknown*. The results are shown in Table 8.15. Here, *PCC* is found as 91.15% with error rate 0.0885.

8.6.3.3 Results using Portscan Dataset

The confusion matrix for all-attacks categories on the *PortscanTrain* dataset is shown in Table 8.16. *Recall* values for *Probe* and *Normal* are respectively 0.9891 and 0.9973 in Table 8.16. 1-*Precision* values for *Probe* and *Normal* are 0.0004 and 0.1497, respectively in Table 8.16. The results indicate that the detection rate of our method is high with low false positives. The average execution time for classification is 0.21 minute.

In cross evaluation for *Portscan* dataset *PortscanTrain* data is used for training and *PortscanTest* data is used for testing. The attack records which exist in *PortscanTest* but do not exist in *PortscanTrain* for *Portscan* dataset are marked as *unknown*. The result is shown in Table 8.17. Here, *PCC* is found as 94.31% with error rate 0.0569.

Table 8.15: Confusion matrix for cross evaluation with *Training* data for training and *Testing* data for testing in *Flow Level*-dataset

Actual	Predicted Class						
Class	*DoS*	*Probe*	*Normal*	*Ukn@*	Sum	Recall	1-Prc*
DoS	9828	153	194	800	10975	0.8955	0.0509
Probe	165	8697	170	448	9480	0.9174	0.0361
Normal	298	147	15305	1020	16770	0.9415	0.0307
Ukn@	64	26	120	3290	3500	0.9400	0.4081
Sum	10355	9023	15789	5558	40725		
Re-substitution error rate=0.0885					PCC=91.15%		

Note- *1-Precision @Unknown

Table 8.16: Confusion matrix for *PortscanTrain* data in Portscan datasets

Actual	Predicted Class				
Class	*Probe*	*Normal*	Sum	Recall	1-Prc*
Probe	38787	428	39215	0.9891	0.0004
Normal	14	2431	2445	0.9973	0.1497
Sum	38801	2859	41660		
Re-substitution error rate=0.0106				PCC=98.94%	

Note- *1-Precision

Table 8.17: Confusion matrix for cross evaluation with *PortscanTrain* data for training and *PortscanTest* data for testing in Portscan dataset

Actual	Predicted Class					
Class	*Probe*	*Normal*	*Unknown*	Sum	Recall	1-Prc*
Probe	12843	206	651	13700	0.9374	0.0032
Normal	15	1253	32	1300	0.9638	0.2065
Unknown	26	120	14117	14915	0.9465	0.0461
Sum	12884	1579	14800	29915		
Re-substitution error rate=0.0569					PCC=94.31%	

Note- *1-Precision

Table 8.18: Comparison of Results for TUIDS dataset *Packet level*

Category	Measure	Baysian Network	Naive Bayes	C4.5	MLH-IDS
DoS	***Recall***	0 9911	0 9891	0 9807	0.9989
	1-Prc*	0.0230	0.0219	0 0124	0.0015
Probe	***Recall***	0 9376	0 8580	0 9530	0.9795
	1-Prc*	0.0212	0 0084	0.0168	0 0213
Normal	***Recall***	0.9750	0.9240	0.9820	0 9973
	1-Prc*	0 0106	0.0145	0.0152	0 0024

Note- *1-Precision

Table 8.19: Comparison of Results for TUIDS dataset *Flow level*

Category	Measure	Baysian Network	Naive Bayes	C4.5	MLH-IDS
DoS	***Recall***	0.9760	0 8540	0.9860	0 9985
	1-Prc*	0.0190	0.0860	0.0140	0.0058
Probe	***Recall***	1.0000	0.9040	0 9950	0 9891
	1-Prc*	0.0050	0.0814	0.0030	0.0271
Normal	***Recall***	0 9550	0.2690	0.9750	0.9864
	1-Prc*	0.0830	0 0120	0 0480	0.0038

Note- *1-Precision

Table 8.20: Comparison of Results for *PortscanTrain* dataset

Category	Measure	Baysian Network	Naive Bayes	C4.5	MLH-IDS
Probe	***Recall***	1.0000	0.9450	0.9760	0.9891
	1-Prc*	0.0150	0.0845	0.0120	0.0004
Normal	***Recall***	0 9650	0 4690	0.9835	0.9973
	1-Prc*	0.0870	0.0070	0.0510	0.1497

Note- *1-Precision

Table 8.21: Comparison of Detection Rate for *Corrected KDD* dataset (%)

Category	Three-level Tree classifier[2]	Multiple-level Hybrid classifier[3]	SVM-based IDS[179]	C4.5[157]	MLH-IDS
DoS	98.54	99.19	99.53	0.9997	**99.99**
Probe	93.50	99.71	97.55	0.9482	98.75
Normal	94.68	96.80	99.29	0.9442	90.07
U2R	97.14	66.67	19.73	0.6711	81.43
R2L	48.91	89.14	28.81	0.8153	**91.10**

8.6.4 Performance Comparisons

For comparison, the performance rates of the decision tree based algorithm *C*4.5[157] and the graph based algorithms *Bayesian Network*[184] and *Naive Bayes*[124] using Weka[228] are given in Table 8.18, 8.19 and 8.20, respectively. The results of MLH-IDS outperform the other three methods in all categories *DoS*, *Probe* and *Normal*.

A summarized comparison of results of MLH-IDS with other competing methods reported in[2,3,179,157] for the KDD Cup 1999 intrusion datasets is reported in Table 8.21. We see in Table 8.21 that MLH-IDS outperforms all other methods for *DoS* and *R2L* categories. In case of *Probe*, the proposed method outperforms all other methods except the method reported in[3]. Similarly, in case of U2R category attack, MLH-IDS outperforms all other methods except the method reported in[2]. However, the performance of the proposed method is not satisfactory in case of normal identification.

8.6.5 Discussion

A multi-level hybrid intrusion detection method is presented based on supervised, unsupervised and outlier methods. The proposed method exhibits good results when used with the TUIDS real life intrusion datasets and two benchmark datasets, viz., KDDCup 1999 and NSL-KDD. Among the methods proposed in *Chapters* 5, 6 and 7, it exhibits excellent classification accuracy for the datasets TUIDS, KDDCup and NSL-KDD as given in Table 8.22. Un-

Table 8.22: Comparison of Classification Accuracy

Dataset	PCC(%)			
	Supervised Method	Unsupervised Method	Outlier-based Method	MLH-IDS
Corrected KDD	97.57	96.08	35.96	98.68
10% Corrected KDD	99.96	95.56	48.00	99.97
KDDTrain$^+$	99.49	95.77	54.00	99.62
KDDTest$^+$	98.39	97.88	36.91	98.82
Packet Level	99.42	99.23	64.25	99.68
Flow Level	99.01	99.43	65.61	99.17
Portscan	98.31	99.16	67.50	98.94

Table 8.23: Average execution time of classification process (expressed in minutes)

Dataset	Supervised Method	Unsupervised Method	Outlier-based Method	MLH-IDS
Corrected KDD	1.00	2.65	1.11	1.36
10% Corrected KDD	1.59	4.21	1.74	1.91
KDDTrain$^+$	0.41	1.07	0.45	0.67
KDDTest$^+$	0.07	0.07	0.07	0.11
Packet Level	0.39	1.04	0.44	0.64
Flow Level	0.17	0.45	0.19	0.25
Portscan	0.14	0.35	0.15	0.21

like the other supervised or unsupervised methods, it can detect both known as well as unknown attacks.

The MLH-IDS method is also useful from the point of view of distributed implementation to achieve of faster in intrusion detection. An important advantage of the MLH-IDS architecture is that it can be configured for any number of known attacks. It helps minimize both space and time complexities for each classifier. A comparison of average execution times of classification by the other methods proposed in *Chapters* 5, 6 and 7 is given in Table 8.23. The MLH-IDS method includes additional communication overhead due to transfer of unclassified data from one classifier to another. However, since all the three classifiers run on a single machine in our experiments, it is negligible.

In *Chapter 9*, we present the conclusion of the thesis and a possible future direction of our research in intrusion detection.

CHAPTER 9

Conclusion and Future work

Contents

9.1 Conclusions

In this thesis, I have reported four significant contributions to detect network anomalies using data mining approaches: supervised detection, unsupervised detection, outlier based rare type anomaly detection and hybrid anomaly detection. The primary concern behind these contributions is to detect both known as well as unknown types of anomalies with high detection accuracy and consequently with low false positives. Accordingly, evaluated each of these contributions using benchmark and public intrusion datasets as well as real life private intrusion datasets.

In supervised anomaly detection, I have contributed an incremental clustering based detection method. The method is trained with labeled training data consisting of normal and anomalous instances. Profiles are created for normal and anomalous classes using similarity measures. We have developed two similarity measures and they are applied for clustering the training data in generation of profile for the said clusters. The proximity measures have been established as a metric in terms of standard properties. The profiles generated are used for classification of test data. The method exhibits very good performance in comparison to its several other counterparts for both public benchmark and private intrusion datasets. The method presented also can be utilized for classification of any other datasets from other domain.

In unsupervised anomaly detection method, a clustering algorithm, *k-point* is developed. The clustering algorithm performs grouping of testing data into normal and anomalous instances on basis of two similarity measures and two assumptions about the data. We have introduced two new similarity measures. The method has been established to be capable of detecting unknown attacks. The effectiveness of this unsupervised method is established over several benchmark and real life intrusion datasets.

In my third contribution, an outlier detection algorithm is developed including an outlier factor using symmetric neighbourhood relationship. The method is applied for anomaly detection and an outlier factor is computed for each instance. On basis of outlier factor and a predefined threshold value for outlier factors, instances are identified as outlier with respect to the rest of the instances. These outlier instances are meant by 'anomalies'. The method is established to be capable of detecting rare class attacks. I have reported the performance of evaluation of this method in detecting the *U2R* and *R2L* attacks on two benchmark datasets and the results are very much comparable with other competing algorithms. The method also has been established to be effective over three real life private datasets.

Classification accuracy of an individual classifier is not equally well for classification of normal instances as well as all categories of attack. There may have a possibility for using an appropriate combination among multiple good performing classifiers to obtain high accuracy classification performance for all categories of attack in a dataset. A multi-level hybrid intrusion detection method (MLH-IDS) is developed in combination of three well performing methods- CatSub[225], *k-point*[226] and outlier detection method[229]. The method performed well for both known and unknown intrusion detection. The method exhibits detection capability for both known and unknown attacks. This method is evaluated with benchmark and captured real time intrusion datasets.

Apart from these four major contributions, I have also reported : (i) An exhaustive survey on existing supervised, unsupervised, probabilistic, data mining, knowledge based and hybrid learning methods. Also, a detailed analysis of these methods in terms of their pros and cons is reported. (ii) Basics of (a) networking, its software and hardware components, (b) anomalies, its sources, types and detection fundamentals. (iii) Fundamentals of various detection approaches, their pros and cons and their evaluations and also the generation of private intrusion datasets.

9.2 Future work

The work contemplated in this thesis can be expanded and improved in a number of different ways. Below, we enlist some of ideas for future work.

- Feature selection is one of the important problem solving methods by reduction of dimensionality in data with selection of features relevant to the subject of searching information. This makes computation for required information faster and comparatively easier. The use of more practical feature selection method in our proposed methods can improve the detection accuracy.

- Cluster labeling solves the problem of classification of clusters, usually, by using the interpretation of physical properties of the clusters. The clustering techniques merely group the data into clusters. The exact or correct labeling of clusters is important for consistent operation of the label in all circumstances. In case of network anomaly detection, the application of clustering method is needed most often for unknown attack detection. Thus, an efficient and faster cluster labeling method is also equally important. An individual labeling method may not be capable of providing correct labeling. Hence, the solution for correct labeling may be with an ensemble approach of labeling methods.

- In rare class attack detection outlier mining method has immense capability. Outliers are the instances which do not obey the rules valid for the majority of the instances. Hence, rare class attacks can be considered as outliers in summarization of all category attacks. A rough set is related to working with a set on its boundary regions An attribute based rough sets technique reduces the complexity of computation of learning processes. It can eliminate the irrelevant or unimportant attributes for efficient learning of experimental or knowledge discovery in database. A fuzzy set theory stipulates to a mathematical framework that represent and treat imprecision, uncertainty, and approximate reasoning. A combined approach of rough set theory and fuzzy set theory can provide a faster and efficient solution for rare category attack detection by finding outliers. Feature selection improves classification by searching for the subset of feature, which best classifies the data. Thus, the rough set techniques can be applied for data cleaning and feature selection. The filtered and feature selected data can be grouped into approximate clusters by use of fuzzy clustering methods.

- Distributed denial-of-service (DDoS) attacks create an extremely large threat to the Internet. One of our future goal is develop a method to combat DDoS attacks by either enhancement of our proposed method or introduce a new one. DDoS attacks are cooperative attacks in large-scale. These attacks are launched from several of compromised hosts. Each compromised host behaves as a legitimate source. This behaviour invalidates many usual anomaly-based network attack detection techniques. The only hope for detecting an effective DDoS attack is possible from early detection of features which an attacker usually can not change or impossible to modify, for instance, the percentage of new IP addresses those seen by the victim machines[230]. We have a plan to work in this direction for the purpose.

- Furthermore, ensemble method for hybrid intrusion detection is another future work. A multi-level hybrid intrusion detection is a combined network anomaly approach for detection of known as well unknown attacks. This approach combines supervised and unsupervised anomaly detection methods in appropriate participation. In its ensemble approach, instead of one supervised and one unsupervised method, more than one well performing supervised and unsupervised methods can be combined for the purpose of high accuracy attack detection. The challenging tasks for ensemble based hybrid intrusion detection method are participation of different methods, their appropriate (unbiased) combination, and faster and high detection accuracy.

- It is also utmost importance to carry out future research in detection of application layer attacks viz., SQL injection, HTTP flooding. XSS attacks etc. These attacks are especially concerned with the vulnerability exploitation in the application layer. Although, in most cases it focuses as intrusion. For example, the causes of these intrusions may b'e the vulnerabilities in the Web servers or browsers.

SQL injection is a common attack issue with web sites of relating to database. The vulnerability of these web sites can be detected easily and can be easily exploited. Thus, any web site or software which have a minimal user base is prompted to be target of this kind of attack.

In cross site scripting (XSS) attack, attackers inject code into the web pages which are created by the vulnerable and malicious web application. On the client side, the attack code is executed with the advantages of the vulnerable web server. Thus, poorly sanitized data is the root cause of XSS vulnerability.

Bibliography

1. Xiang, C., et al. Design of multiple-level tree classifiers for intrusion detection system. In *Proceeding of 2004 IEEE Conference on Cybernetics and Intelligent Systems*, 872–877 (IEEE, Singapore, 2004). (Cited on pages xx, 128, 235 and 240.)

2. Lu, H. & Xu, J. Three-Level hybrid intrusion detection system. In *Proceedings of International Conference on Information Engineering and Computer Science, ICIECS 2009*, 1–4 (IEEE, Shanghai, China, 2009). (Cited on pages xxi, 128, 235, 236, 240 and 256.)

3. Xiang, C., et al. Design of multiple-level hybrid classifier for intrusion detection system using Bayesian clustering and decision trees. *Pattern Recognition Letters* **29**(7), 918–924 (2008). (Cited on pages xxi, 236, 237, 240 and 256.)

4. Chen, T.-S., et al. Integrated multilevel intrusion detection and report system. In *Proceedings of the Fifth International Conference on Electronic Business*, 463–469 (ICEB,NTIT, Hong Kong, 2005). (Cited on pages xxi, 236, 237 and 240.)

5. Al-Nashif, Y., et al. Multi-Level Intrusion Detection System (ML-IDS). In *Proceedings of the 2008 International Conference on Autonomic Computing (ICAC '08)*, 131–140 (IEEE Computer Society, Chicago, Illinois, USA, 2008). (Cited on pages xxi, 237, 238 and 240.)

6. Thottan, M. & Ji, C. Anomaly Detection in IP Networks. *IEEE Transactions on Signal Processing* **51**(8), 2191–2204, August (2003). (Cited on pages 4 and 46.)

7. Wu, Z., et al. A Taxonomy of Network and Computer Attacks Based on Responses. In *Proceedings of the International Conference of Information Technology, Computer Engineering and Management Sciences*, 26–29 (IEEE Computer Society, China, 2011). (Cited on pages 5 and 53.)

8. Chandola, V., et al. Anomaly Detection : A Survey. *ACM Computing Surveys (CSUR)* **41**(3), 15:1–15:58, July (2009). (Cited on pages 5, 50, 62 and 207.)

9. Han, J. & Kamber, M. *Data Mining : Concepts and Techniques*. Simon Fraser University, Morgan Kaufmann Publishers, (2006). (Cited on pages 5 and 6.)

10. Tan, P.-N., et al. *Introduction to Data Mining*. Pearson Education, Inc., (2009). (Cited on page 5.)

11. Hand, D., et al. *Principles of Data Mining*. Prentice-Hall of India, New Delhi, (2004). (Cited on pages 5 and 6.)

12 Eskin, E. Anomaly detection over noisy data using learned probability distributions. In *Proceedings of the 7th International Conference on Machine Learning (ICML-2000)*, 255–262 (Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2000). (Cited on pages 8, 102 and 190.)

13. Eskin, E., et al. Modeling system call for intrusion detection using dynamic window sizes. In *Proceedings of DARPA Information Survivability Conference and Exposition (DISCEX II'01)*. IEEE Computer Society, (2001). (Cited on pages 8 and 102.)

14. Ye, T., et al. Network Management and Control Using Collaborative On-line Simulation. In *Proceedings of IEEE International Conference on Communications (ICC 2001)* (IEEE Computer Society Press, Los Alamintos, CA, Helsinki, Finland, 2001). (Cited on pages 8 and 101.)

15. Gwadera, R., et al. Detection of significant sets of episodes in event sequences. In *Proceedings of the 4th IEEE International Conference on Data Mining*, 3–10 (IEEE Computer Society, Washington, DC, USA, 2004). (Cited on pages 8 and 101.)

16 Gwadera, R., et al. Reliable detection of episodes in event sequences. *Knowledge and Information Systems* **7**(4), 415–437 (2005). (Cited on pages 8 and 101.)

17. Sequeira, K. & Zaki, M. ADMIT: Anomaly-based Data Mining foi Intrusions. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 386–395, (2002). (Cited on pages 8 and 104.)

18. Burbeck, K. & Nadjm-Tehrani, S. ADWICE - Anomaly Detection with Real-Time Incremental Clustering In *Proceedings of Information Security and Cryptology -ICISC 2004*, volume 3506/2005, 407–424 (Springer, Berlin, Germany, 2005). (Cited on pages 8, 104, 128, 142 and 143.)

19. Yang, H., et al. Clustering and Classification Based Anomaly Detection. *Fuzzy Systems and Knowledge Discovery* **4223/2006**, 1082–1091 (2006). (Cited on pages 8, 104 and 142.)

20. Lee, W. & Stolfo, S. J. Data mining approaches for intrusion detection. In *Proceedings of the 7th conference on USENIX Security Symposium -*

Volume 7, 1–16 (USENIX, San Antonio, Texas, USA, 1998). (Cited on pages 8, 108, 132, 141 and 230.)

21. Lee, W., et al. A Data Mining Framework for Building Intrusion Detection Models. In *Proceedings of the IEEE Symposium on Security and Privacy*, 120–132 (IEEE, Oakland, CA, USA, 1999). (Cited on pages 8 and 108.)

22. Barbara, D., et al. ADAM: a testbed for exploring the use of data mining in intrusion detection. *SIGMOD Record* **30**(4), 15–24 (2001). (Cited on pages 8, 108, 109, 128, 186, 207 and 231.)

23. Mahoney, M. V. Network Traffic Anomaly Detection Based on Packet Bytes. In *Proceedings of the 2003 ACM Symposium on Applied Computing (SAC)*, 346–350 (ACM, Melbourne, Florida, USA, 2003). (Cited on pages 8 and 108.)

24. Krügel, C., et al. Bayesian event classification for intrusion detection. In *Proceedings of the 19th Annual Computer Security Applications Conference (ACSAC 2003), 8-12 December 2003*, 14–23 (IEEE Computer Society, Las Vegas, NV, USA, 2003). (Cited on pages 8, 112, 113 and 231.)

25. Valdes, A. & Skinner, K. Adaptive Model-based Monitoring for Cyber Attack Detection. In *Proceedings of the Recent Advances in Intrusion Detection*, number 1907 in Lecture Notes in Computer Science, 80–92 (Springer-Verlag, Toulouse, France, 2000). (Cited on pages 8, 112 and 113.)

26. Ye, N. A Markov Chain Model of Temporal Behavior for Anomaly Detection. In *Proceedings of the 2000 IEEE Workshop on Information Assurance and Security*, 171–174 (IEEE Xplore, United States Military Academy, West Point, NY, 2000). (Cited on pages 8 and 112.)

27. Ahirwar, D. K., et al. Anomaly Detection by Naive Bayes & RBF Network. *International Journal of Advanced Research in Computer Science and Electronics Engineering* **1**(1), 14–18 (2012). (Cited on pages 8 and 113.)

28. Ghahramani, Z. An Introduction to Hidden Markov Models and Bayesian Networks. *International Journal of Pattern Recognition and Artificial Intelligence* **15**(1), 9–42 (2001). (Cited on pages 8 and 110.)

29. Tapiador, J. M. E., et al. Detection of Web-based attacks through Markovian protocol parsing. In *Proceedings of the 10th IEEE Symposium on Computers and Communications*, 457–462 (IEEE Computer Society, Madrid, Spain, 2005). (Cited on page 8.)

30. Reynolds, D. A., et al. Speaker Verification Using Adapted Gaussian Mixture Models. *Digital Signal Processing* **10**(1-3), 19–41 (2000). (Cited on pages 8 and 114.)

31. Bahrololum, M. & Khaleghi, M. Anomaly Intrusion Detection System Using Gaussian Mixture Model. In *Proceedings of the 3rd International Conference on Convergence and Hybrid Information Technology*, volume 1, 1162–1167 (IEEE Computer Society, Tehran, 2008). (Cited on pages 8, 115 and 128.)

32. Patcha, A. & Park, J.-M Detecting Denial-of-Service Attacks with Incomplete Audit Data. In *Proceedings of the 14th International Conference on Computer Communications and Networks (ICCCN 2005)*, 263–268 (IEEE Computer Society, Blacksburg, VA, USA, 2005). (Cited on pages 8, 117 and 128.)

33. Zadeh, L. A. Fuzzy logic, neural networks, and soft computing. *Communications, ACM* **37**(3), 77–84, March (1994). (Cited on pages 9, 117, 119 and 186)

34. Bayes, T. An essay towards solving a problem in the doctrine of chances. *Philosophical Transactions of the Royal Society of London* **53**(6), 370–418 (1763). (Cited on pages 9, 110, 117 and 137.)

35 Rosenblatt, F. Two theorems of statistical separability in the perceptron. In *Proceedings of the Symposium of Mechanization of Thought Processes*, 421–456 (National Physical Laboratory, HM Stationary Office, London, 1959). (Cited on pages 9 and 117.)

36. Holland, J. H Adaptation in natural and artificial systems. MIT Press, Cambridge, MA, (1975). (Cited on pages 9 and 117.)

37. Zadeh, L. A. Role of Soft Computing and Fuzzy Logic in the Conception, Design and Development of Information/Intelligent Systems *Lecture Notes in Computer Science* **695**, 1–9 (1998). (Cited on pages 9, 117 and 189.)

38 Haykin, S. *Neural Networks*. Prentice Hall, New Jersey, (1999). (Cited on pages 9 and 118.)

39. Li, Y.-Z., et al. Anomaly Intrusion Detection Method Based on Rough Set Theory. In *Proceedings of the International Conference on Wavelet Analysis and Pattern Recognition* (IEEE Computer Society, Hong Kong, 2008). (Cited on pages 9, 119 and 128.)

40. Chris, S., et al. An application of machine learning to network intrusion detection. In *Proceedings of the 1999 Annual Computer Security Applications Conf. (ACSAC)*, 371–377 (IEEE Computer Society Press, Phoenix,Arizona, 1999). (Cited on pages 9, 121 and 128.)

41. Denning, D. E. & Neumann, P. G. Requirements and Model for IDES-A Real-time Intrusion Detection System. Technical Report 83F83-01-00, Computer Science Laboratory, SRI International, Menlo Park, CA, USA (1985). (Cited on pages 9 and 122.)

42. Lunt, T. F., et al. A Real-time Intrusion Detection Expert System (IDES). Technical report, Computer Science Laboratory, SRI International, Menlo Park, CA, USA, (1992). (Cited on pages 9 and 122.)

43. Shon, T. & Moon, J. A Hybrid Machine Learning Approach to Network Anomaly Detection. *Information Science* **177**, 3799–3821 (2007). (Cited on pages 10, 66, 123, 128 and 190.)

44. Bahrololum, M., et al. Anomaly Intrsion Detection Design Using Hybrid of Unsupervised and supervised Neural Network. *International Journal of Computer Networks & Communications (IJCNC)* **1**(2), 26–33, July (2009). (Cited on pages 10, 66, 123 and 125.)

45. Yang, J., et al. HIDS-DT: An Effective Hybrid Intrusion Detection System Based on Decision Tree. In *Proceeding of the International Conference on Communications and Mobile Computing*, 70–75. IEEE Computer Society, (2010). (Cited on pages 10, 123 and 126.)

46. Zhang, J. & Zulkernine, M. A Hybrid Network Intrusion Detection Technique using Random Forests. In *Proceeding of 1st International Conference on Availability, Reliability and Security (ARES 2006)*, 262–269 (IEEE Computer Society, Vienna, Austria, 2006). (Cited on pages 10, 66, 123, 126, 128, 234 and 240.)

47 Yao, Y., et al. Anomaly Intrusion Detection Approach Using Hybrid MLP/CNN Neural Network. In *Proceedings of the 6th International Conference on Intelligent Systems Design and Applications (ISDA '06)*, 1095–1102 (IEEE Computer Society, Washington, DC, USA, 2006). (Cited on pages 10, 66 and 123.)

48. Garcia-Teodoro, et al. Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers & Security* **28**, 18–28 (2009). (Cited on pages 10 and 142.)

49. Stallings, W. *Data and Computer Communication.* Pearson Prentice Hall, eighth edition, (2007). (Cited on page 17.)

50. Tanenbaum, A. S. *Computer Networks.* Pearson Prentice Hall, fourth edition, (2003). (Cited on pages 17 and 22.)

51. Cerf, V. & Khan, R. A Protocol for Packet Network Interconnection. *IEEE Transactions on Communication* **COM-22**, 637–648, Month (1974). (Cited on page 25.)

52. Stouffer, K. A , et al. Guide to Industrial Control Systems (ICS) Security. Technical Report SP 800-82, National Institute of Standards & Technology, Gaithersburg, MD, United States, (2011). (Cited on page 47.)

53. Anderson, J. P. Computer security threat monitoring and surveillance Technical report, James P Anderson Co., Fort Washington, Pennsylvania, (April,1980). (Cited on pages 51 and 185.)

54. Hansman, S. & Hunt, R. A Taxonomy of Network and Computer Attacks. *Computers & Security* **24**(1), 31–43, September (2005) (Cited on pages 53, 72 and 73.)

55. Neumann, P. G. & Parker, D. B. A Summary of Computer Misuse Techniques. In *Proceedings of the 12th National Computer Security Conference*, 396–407 (CERIAS, Baltimore, MD, 1989). (Cited on page 53.)

56. Lindqvist, U. & Jonsson, E. How to Systematically Classify Computer Security Intrusions. *IEEE Security and Privacy*, 154–163 (1997). (Cited on page 53.)

57. MIT Lincoln Lab., Information Systems Technology Group. The 1998 intrusion detection off-line evaluation plan http://www.11.mit.edu/IST/ideval/docs/1998/id98-eval-11.txt, (1998). (Cited on pages 54, 72 and 73.)

58. Marinova-Boncheva, V. A short survey of intrusion detection systems. *Institute of Information Technologies* **1113 Sofia**, 23–30 (2007). (Cited on pages 54 and 55.)

59. Kumar, V., et al. Managing cyber threats issues, approaches, and challenges. *Massive Computing* **5**(XVIII), 1–330, June (2005). (Cited on pages 55 and 56.)

60. Bhuyan, M. H., et al. Surveying Port Scans and Their Detection Methodologies. *The Computer Journal)* **54**(4), 1–17, April (2011). (Cited on page 56.)

61. Gogoi, P., et al. A Survey of Outlier Detection Methods in Network Anomaly Identification. *The Computer Journal* **54**(4), 570–588 (2011). (Cited on pages 60, 68, 69, 107, 208 and 234.)

62. Joshi, M. V., et al. Mining needle in a haystack: classifying rare classes via two-phase rule induction. In *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 293–298. ACM Press, (2001). (Cited on pages 61, 88 and 89.)

63. Theiler, J. & Cai, D. M. Resampling approach for anomaly detection in multispectral images. In *Proceedings of SPIE*, volume 5093, 230–240. SPIE, (2003). (Cited on pages 61 and 100.)

64. Fujimaki, R., et al. An approach to spacecraft anomaly detection problem using kernel feature space. In *Proceeding of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, 401–410 (ACM Press, New York, USA, 2005). (Cited on page 61.)

65. Leon, E., et al. Anomaly Detection Based on Unsupervised Niche Clustering with Application to Network Intrusion Detection. *IEEE Congres on Evolutionary Computation* **1**, 502–508 (2004). (Cited on pages 64 and 232.)

66. Leung, K. & Leckie, C. Unsupervised Anomaly Detection in Network Intrusion Detection Using Clusters. In *Proceedings of 28th Australasian Conference on Computer Science - Volume 38*, 333–342 (Australian Computer Society, Inc. Darlinghurst, Newcastle, NSW, Australia, 2005). (Cited on pages 64, 190, 203 and 232.)

67. Chimphlee, W., et al. Unsupervised Clustering Methods for Identifying Rare Events in Anomaly Detection. In *Proceedings of World Academy of Science, Engineering and Technology (PWASET)*, volume 8, 253–258 (WASET, Stankin, Moscow, 2005). (Cited on pages 64 and 232.)

68. Zhong, S., et al. Clustering-based network intrusion detection. *International Journal of Reliability, Quality and Safety Engineering* **14**(2) (2007). (Cited on pages 64 and 232.)

69. Zhang, C., et al. A mixed unsupervised clustering-based intrusion detection. In *Proceedings of 3rd International Conference on Genetic and Evolutionary Computing, WGEC 2009* (IEEE Computer Society, Gulin, China, 2009). (Cited on pages 64 and 232.)

70. Portnoy, L., et al. Intrusion Detection with Unlabeled Data using Clustering. In *Proceedings of ACM CSS Workshop on Data Mining Applied to Security (DMSA-2001)*, 1–14 (ACM, Philadelphia, PA, USA, 2001). (Cited on pages 64, 66, 88, 104, 187, 230 and 232.)

71. Storløkken, R. Labelling clusters in an anomaly based ids by means of clustering quality indexes. Master's thesis, Faculty of Computer Science and Media Technology Gjøvik University College, (2007). (Cited on page 65.)

72. Dunn, J. Well separated clusters and optimal fuzzy partitions. *Journal of Cybernetics* **4**, 95–104 (1974). (Cited on pages 65, 239, 240 and 245.)

73. Hubert, L. & Schultz, J. Quadratic assignment as a general data-analysis strategy. *British Journal of Mathematical and Statistical Psychology* **29**, 190–241 (1976). (Cited on page 65.)

74. Davies, D. L. & Bouldin, D. W. A cluster separation measure. *IEEE Transaction on Pattern Analysis and Machine* **1**(2), 224–227 (1979). (Cited on pages 65, 240, 241 and 245.)

75. Rousseeuw, P. J. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics* **20**(1), 53–65 (1987). (Cited on pages 65, 240, 241 and 245.)

76. Xie, X. L. & Beni, G. A validity measure for fuzzy clustering. *IEEE Transaction on Pattern Analysis and Machine Intelligence* **13**(4), 841–847 (1991). (Cited on pages 65, 240, 241 and 245.)

77. Cha, S. H. Comprehensive survey on distance/similarity measures between probability density functions. *International Journal of Mathematical Models and Methods in Applied Science* **1**(4), Nov. (2007). (Cited on pages 67, 68 and 145.)

78. Boriah, S., et al. Similarity measures for categorical data: A comparative evaluation. In *Proceedings of the 8th SIAM International Conference on Data Mining*, 243–254 (Society for Industrial and Applied Mathematics(SIAM), Atlanta, Georgia, USA, 2008). (Cited on pages 68, 145 and 188.)

79. Koufakou, A. & Georgiopoulos, M. A fast outlier detection strategy for distributed high-dimensional data sets with mixed attributes. *Data Mining and Knowledge Discovery* **20**(2), 259–289, Mar. (2010). (Cited on pages 68, 107, 209 and 234.)

80. Amiri, F., et al. Mutual information-basedfeatureselectionforintrusiondetectionsystems. *Journal of Network and Computer Applications* **34**, 1184–1199 (2011). (Cited on pages 69 and 155.)

81. University of California. KDD Cup 1999 Data. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, (1999). (Cited on pages 71, 158, 192, 198, 213, 219 and 247.)

82. Tavallaee, M., et al. A Detailed Analysis of the KDD CUP 99 Data Set. In *Proceedings of the 2009 IEEE Symposium on Computational Intelligence in Security and Defence Applications (CISDA 2009)*, 1–6 (IEEEXplore, Ottawa, Canada, 2009). (Cited on pages 74, 158, 192, 198, 213, 219 and 247.)

83. Mixter, D. Attacks tools and information. http://packetstormsecurity.nl/index.html, (2003). (Cited on page 75.)

84. Amini, M., et al. RT-UNNID: A practical solution to real- time network-based intrusion detection using unsupervised neural networks. *Computers & Security* **25**(6), 459–468 (2006). (Cited on pages 75, 189, 223, 224 and 245.)

85. Satten, C. Lossless gigabit remote packet capture with Linux. University of Washington Network Systems, http://staff.washington.edu/corey/gulp/, (March, 2008). (Cited on page 81.)

86. Combs, G. Wireshark. http://www.wireshark.org/, (2009). (Cited on page 81.)

87. Ostermann, S. Tcptrace. Ohio University, http://www.tcptrace.org, (2009). (Cited on page 81.)

88. Quittek, J., et al. RFC 3917: Requirements for IP Flow Information Export: IPFIX, Hawthorn Victoria. http://www.ietf.org/rfc/rfc3917.txt, (2004). (Cited on page 81.)

89. Claise, B. RFC 3954: Cisco Systems NetFlow Services Export Version 9. http://www.ietf.org/rfc/rfc3954.txt, (2004). (Cited on page 81.)

90. Haag, P. Nfdump & nfsen. http://nfdump.sourceforge.net/, (September, 2010). (Cited on page 84.)

91. Cisco.com. Cisco IOS NetFlow Configuration Guide, Release 12.4. http://www cisco.com, (September, 2010). (Cited on page 84.)

92. Gogoi, P., et al. Packet and Flow Based Network Intrusion Dataset. In *Proceedings of the 5th International Conference on Contemporary Computing (IC3-2012)*, volume 306 of *CCIS*, 322–334 (Springer, Noida, India, 2012). (Cited on pages 85, 158, 198, 213, 219, 231 and 247.)

93. Axelsson, S. The base-rate fallacy and its implications for the difficulty of intrusion detection. 1–7 (ACM New York, NY, USA, 1999). (Cited on pages 88 and 93.)

94. Axelsson, S. The base-rate fallacy and the difficulty of intrusion detection. *ACM Transactions on Information and System Security (TISSEC)* **3**(3), 186–205 (2000). (Cited on pages 88 and 93.)

95. Ertoz, L., et al. Detection of novel network attacks using data mining. In *Proceedings of the 2003 ICDM Workshop on Data Mining for Computer Security*, 30–39, (2003). (Cited on page 88.)

96. Tavallaee, M., Stakhanova, N., & Ghorbani, A. A. Towards credible evaluation of anomaly-based intrusion detection methods. *IEEE Transactions on System, Man, and Cybernetics Part C: Applications and Reviews* **40**(5), 516–524, Sep. (2010). (Cited on pages 89 and 93.)

97. van Rijsbergen, C. J. *Information Retrieval.* MA: Butterworth, London, GB, Boston, 2nd edition, (1979). (Cited on page 91.)

98. Maxion, R. A. & Townsend, T. N. Masquerade detection augmented with error analysis. *IEEE Transactions on Reliability* **53**(1), 124–147 (2004). (Cited on pages 92 and 93.)

99. Jacobson, D. *Introduction to Network security.* Chapman and Hall/CRC Press, (2011). (Cited on page 100.)

100. Yu, Z. & Tsai, J. J. P. *Intrusion Detection- a Machine Learning Approach.* Imperial College Press, (2011). (Cited on page 100.)

101. Wang, Y. *Statistical Technique for Network Security.* Information Science Reference (IGI Global), (2009). (Cited on page 100.)

102. Ghorbani, A., Lu, W., & Tavallaee, M. *Network Intrusion Detection and Prevention-Concepts and Techniques.* Prentice-Hall (PTR), (2009). (Cited on page 100.)

103. Barnett, V. & Lewis, T. *Outliers in Statistical Data.* John Wiley, Chichester, New York, USA, (1994). (Cited on pages 101, 107 and 191.)

104. Montgomery, D. C. *Introduction to Statistical Quality Control.* John Wiley & Sons, New York, USA, (2000). (Cited on page 102.)

105. Banks, J. *Principles of Quality Control.* John Wiley & Sons, New York, USA, (1989). (Cited on page 102.)

106. Yamanishi, K. & ichi Takeuchi, J. Discovering outlier filtering rules from unlabeled data: combining a supervised learner with an unsupervised learner. In *Proceedings of the 7th ACM SIGKDD international conference on Knowledge discovery and data mining*, 389–394 (ACM Press, San Francisco, California, USA, 2001). (Cited on page 102.)

107. Yamanishi, K., et al. On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms. *Data Mining and Knowledge Discovery* **8**, 275–300 (2004). (Cited on page 102.)

108. Mahoney, M. V. & Chan, P. K. Learning nonstationary models of normal network traffic for detecting novel attacks. In *Proceedings of the 8th ACM SIGKDD international conference on Knowledge discovery and data mining*, 376–385 (ACM Press, Edmonton, Alberta, Canada, 2002). (Cited on page 102.)

109. Zhao, M. & Saligrama, V. Anomaly Detection with Score functions based on Nearest Neighbor Graphs. In *Proceedings of the 23rd Annual Conference on Neural Information Processing Systems (NIPS)*, volume 22, 2250–2258 (Curran Associates, Inc., Vancouver, British Columbia, Canada, 2009). (Cited on page 103.)

110. Zanero, S. & Savaresi, S. M. Unsupervised learning techniques for an intrusion detection system. In *Proceedings of the 2004 ACM symposium on Applied computing*, 412–419, (2004). (Cited on page 104.)

111. Casas, P., et al. UNADA: Unsupervised Network Anomaly Detection using Sub-Space Outliers Ranking. In *Proceedings of the 10th international IFIP TC 6 conference on Networking - Volume Part I (NETWORKING'11)*, 40–51 (Springer-Verlag Berlin, Heidelberg, 2011). (Cited on pages 105 and 128.)

112. Ertöz, L., et al. The MINDS - Minnesota intrusion detection system. In *Next Generation Data Mining* (MIT Press, Boston, 2004). (Cited on pages 107, 128 and 143.)

113. Ghoting, A., et al. Loaded: Link-based Outlier and Anomaly Detection in Evolving Data Sets. In *Proceedings of the 4th IEEE International Conference on Data Mining*, 387–390 (IEEE Computer Society, Brighton, UK, 2004). (Cited on pages 107, 209 and 234.)

114. Breunig, M. M., et al. Lof: Identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on management of data*, 93–104 (New York: ACM Press, Dallas, Texas, United States, 2000). (Cited on pages 107, 108, 209, 213 and 234.)

115. Agrawal, R., et al. Mining association rules between sets of items in large databases. In *Proceedings of the ACM SIGMOD Conference*, 207–216 (ACM, Washington D.C., USA, 1993). (Cited on page 108.)

116. Agrawal, R. & Srikant, R. Mining sequential patterns In *Proceedings of the 11th International Conference on Data Engineering*, 3–14 (IEEE Computer Society, Washington, DC,USA,, 1995). (Cited on page 108.)

117. Hipp, J., et al. Algorithms for Association Rule Mining - a General Survey and Comparison. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 58–64 (ACM, Boston, MA, USA, 2000). (Cited on page 108.)

118. Dempster, A. P. Upper and lower probabilities induced by a multivalued mapping. *Ann. Math. Stat.* **38**, 325–339 (1967). (Cited on page 110.)

119. Shafer, G. A mathematical theory of evidence. Princeton University Press, NJ, (1976). (Cited on page 110.)

120. Ye, N. A Markov Chain Model of Temporal behavior for Anomaly Detection. In *Proceedings of the 2000 IEEE Workshop on Information Assurance and Security United States Military Academy*, 171–174 (IEEE, West Point, NY, USA, 2000). (Cited on page 111.)

121. Yeung, D. Y. Host-based Intrusion Detection Using Dynamic and Static Behavioral Models. *Pattern Recognition* **36**, 229–243 (2003) (Cited on pages 112 and 128.)

122. Heckerman, D A tutorial on learning with Bayesian networks. Technical report, Microsoft Research, MSRTR-95-06., (1995). (Cited on page 112)

123. Barbara, D., et al. Detecting novel network intrusions using bayes estimators In *Proceedings of the First SIAM International Conference on Data Mining.*, (2001). (Cited on page 112.)

124. Langley, P., et al. An analysis of Bayesian classifiers. In *Proceedings of the Tenth National Conference of Artificial Intelligence*, 223–228. AAAI Press, (1992). (Cited on pages 113, 236 and 256.)

125. Park, J..& Sandberg, J. W. Universal Approximation using radial basis functions network. *Neural Computation* **3**, 246–257 (1991) (Cited on page 113.)

126. Porras, P. A. & Neumann, P. G. Emerald: Event monitoring enabling responses to anomalous live disturbances. Technical report, SRI International, Menlo Park, CA 94025, (1997). (Cited on page 113.)

127. Dempster, A., et al. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal Royal Statistical Society* **39**(1), 1–38 (1977). (Cited on page 116.)

128. Bilmes, J. A gentle tutorial on the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. Technical Report ICSI-TR-97-021, University of Berkeley, (1997). (Cited on page 116.)

129. Ryan, J., et al. Intrusion Detection with Neural Networks. *Advances in Neural Information Processing Systems* **10** (1998). (Cited on pages 118 and 128.)

130. Pawlak, Z., et al. Rough sets. *Communications of the ACM* **38**(11), 88–95, Nov. (1995). (Cited on pages 119 and 210.)

131. Sarkar, M. & Yegnanarayana, B. Fuzzy-rough membership functions. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, volume 2, 2028–2033 (IEEE Xplore, San Diego, CA , USA, 1998). (Cited on pages 119 and 210.)

132. Gogoi, P., et al. Efficient Rule Set Generation using Rough Set Theory for Classification of High Dimensional Data. *International Journal of Smart Sensors and Ad Hoc Networks (IJSSAN)* **1**(2), 13–20 (2011). (Cited on page 119.)

133. Dickerson, J. E. & Dickerson, J. A. Fuzzy network profiling for intrusion detection. In *Proceedings of NAFIPS 19th International Conference of the North American Fuzzy Information Processing Society*, 301–306 (IEEE, Atlanta, USA, 2000). (Cited on pages 120, 128 and 143.)

134. Noel, S., et al. Modern intrusion detection, data mining, and degrees of attack guilt. In *Applications of Data Mining in Computer Security*. Springer, (2002). (Cited on page 122.)

135. Anderson, D., et al. Next-generation intrusion detection expert system (nides) :a summary. Technical report, Computer Science Laboratory SRI-CSL-95-07, (1995). (Cited on pages 122 and 128.)

136. Glaser, T. *TCP/IP Stack Fingerprinting Principles*. SANS, (2000). (Cited on page 123.)

137. Cho, S.-B. Ensemble of Structure-adaptive Self-organizing Maps for High Performance Classification. *Information Sciences* **123**, 103–114 (2000). (Cited on page 123.)

138. Corinna, C & Vapnik, V. Support-vector Network. *Machine Learning* **20**, 273–297 (1995). (Cited on page 124.)

139. Heller, K. A., et al. One Class Support Vector Machines for Detecting Anomalous Windows Registry Accesses. In *Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM)*, 281–289 (IEEE Computer Society Press, Melbourne, FL, 2003). (Cited on page 124.)

140. Kohonen, T. *Self-Organizing Maps.* Springer, Berlin, (2000). (Cited on page 125.)

141. Rumelhart, D., et al. Learning Internal Representations by Error Backpropagation. In *D. Rumelhart, J. McClelland and The PDP Research Group (Ed.) Parallel Distributed Processing: Explorations into the Microstructure of Cognition*, 318–362 (The MIT Press, Cambridge, 1986). (Cited on page 125.)

142. Molloy, M. K. Performance Analysis Using Stochastic Petri Nets. *IEEE Transactions on Computers* **C-31**(9), 913–917, September (1982). (Cited on page 126.)

143. Marsan, M. A., et al. Modelling with Generalized Stochastic Petri nets. *ACM SIGMETRICS Performance Evaluation Review* **26**(2), 2–3 (1998). (Cited on page 126.)

144. Roesch, M. Snort-lightweight intrusion detection for networks. In *Proceedings of the 13th USENIX conference on System administration*, 229–238 (USENIX, Seattle, Washington, 1999). (Cited on pages 128, 136, 150, 186 and 207.)

145. Labib, K & Vemuri, R. NSOM: A Tool To Detect Denial Of Service Attacks Using Self-Organizing Maps. Technical report, Department of Applied Science University of California, Davis, California, U.S.A., (2002). (Cited on pages 128 and 143.)

146. Mohajerani, M., et al. NFIDS: A Neuro-Fuzzy Intrusion Detection System. In *Proceedings of the 10th IEEE International Conference on Electronics, Circuits and Systems(ICECS)*, volume 1, 348–351, , December (2003). (Cited on pages 128 and 207.)

147. Zhang, Z., et al. HIDE: a Hierarchical Network Intrusion Detection System Using Statistical Preprocessing and Neural Network Classification. In *Proceedings of 2001 IEEE Man Systems and Cybernetics Information Assurance Workshop*, (2001). (Cited on page 128.)

148. Sequeira, K. & Zaki, M. ADMIT: anomaly-based data mining for intrusions. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 386–395 (ACM Press, Edmonton, Alberta, Canada, 2002). (Cited on page 128.)

149. Kuang, L. V. DNIDS: A Dependable Network Intrusion Detection System Using the CSI-KNN Algorithm. Master's thesis, Queen's University Kingston, Ontario, Canada, , September (2007). (Cited on page 128.)

150. Otey, M E., et al. Fast lightweight outlier detection in mixed-attribute data. Technical report, Department of Computer Science and Engineering, The Ohio State University, Ohio, United States, (2005). (Cited on page 128.)

151. Duan, L., et al. Cluster-based outlier detection. *Annals of Operations Research* **168**(1), 151–168, June (2008). (Cited on page 128.)

152. Depren, O., et al. An intelligent intrusion detection system (IDS) for anomaly and misuse detection in computer networks. *Expert Systems with Applications* **29**(4), 713–722 (November 2005). (Cited on pages 128, 234 and 240.)

153. Hwang, K., et al. Hybrid Intrusion Detection with Weighted Signature Generation Over Anomalous Internet Episodes. *IEEE Transactions on Dependable and Secure Computing* **4**(1), 41–55 (2007). (Cited on pages 128, 235 and 240.)

154. Brieman, L., et al. Classification and reregression trees. Wadsworth & Brooks, Monterey, CA, (1984). (Cited on pages 133 and 180.)

155. Gini, C. Variabilita e Mutabilita. *Journal of the Royal Statistical Society* **76**(3), 326-327, February (1913). (Cited on page 134.)

156. Chebrolu, S., et al. Feature Deduction and Ensemble Design of Intrusion Detection Systems. *Computers & Security* **24**(4), 295–307 (2005). (Cited on page 134.)

157. Quinlan, J. R. C4.5: Programs for Machine Learning. Morgan Kaufman, (1993). (Cited on pages 135, 138, 180, 236 and 256.)

158. Pateriya, R. K. & Johar, A. An Intelligent Intruder Detection System Based On Process Mining and Decision Tree. *VSRD International Journal of Computer Science & Information Technology* **2**(9), 782–789 (2012). (Cited on page 135.)

159. Jensen, F. V. Introduction to Bayesian Networks. Springer-Verlag, New York, USA, (1996). (Cited on page 136.)

160. Tylman, W. .Anomaly-Based Intrusion Detection Using Bayesian Networks. In *Proceedings of the 2008 3rd International Conference on Dependability of Computer Systems DepCoS-RELCOMEX (DEPCOS-RELCOMEX '08)*, 211–218 (IEEE Computer Society, Washington, DC, USA, 2008). (Cited on page 136.)

161. Amor, N. B., et al. Naive Bayes vs Decision Trees in Intrusion Detection Systems In *Proceedings of the ACM Symposium on Applied Computing (SAC'04)*, 420–424 (ACM, Nicosia, Cyprus, 2004). (Cited on pages 137 and 231.)

162. Panda, M. & Patra, M. R. Network Intrusion Detection Using Naive Bayes. *International Journal of Computer Science and Network Security* **7**(12), 258–263, December (2007). (Cited on page 138.)

163. Clark, P. & Niblett, T. The CN2 induction algorithm. *Machine Learning* **3**, 261–283 (1989). (Cited on pages 138 and 180.)

164. Michalski; R. S. On the quasi-minimal solution of the general covering problem. In *Proceedings of the Fifth International Symposium on Information Processing (FCIP 1969)*, volume A3, 125–128 (Instytut Automatyki, Bled, Yugoslavia, 1969). (Cited on page 138.)

165. Kalbfieish, J. Probability and statistical inference (vol 2). Springer-Verlag, New York USA., (1979) (Cited on page 139.)

166. Beghdad, R. Critical Study of Supervised Learning Techniques in Predicting Attacks. *Information Security Journal: A Global Perspective* **19**, 22–35 (2010). (Cited on pages 139, 143 and 180.)

167. Hsu, C. W., et al. A practical guide to support vector classification. Technical report, University of Freiburg, (July, 2003). (Cited on pages 140, 189 and 231.)

168. Vapnik, V. N. *The Nature of Statistical Learning Theory*. Springer-Verleg, New York, USA, (1995). (Cited on pages 140, 189, 206 and 231.)

169. Arfken, G. Lagrange Multipliers. In *Mathematical Methods for Physicists 3rd ed*, 945–950 (Academic Press, Orlando, 1985) (Cited on page 140.)

170. Kim, D. S. & Park, J. S. Network-Based Intrusion Detection with Support Vector Machines. In *Proceedings of the International Conference on Information Networking (ICOIN 2003)*, 747–756, (2003). (Cited on page 141.)

171. Agarwal, R. & Joshi. M. V. PNrule: A New Framework for Learning Classifier Models in Data Mining (A Case-Study in Network Intrusion Detection). Technical Report TR 00-015, Department of Computer Science, University of Minnesota, (2000): (Cited on page 141.)

172. Zhang, T., et al. BIRCH: an effective data clustering method for very large databases. *SIGMOID Record 1996 ACM SIGMOID International Conference on Management of Data* **25**, 103–114 (1996). (Cited on page 142.)

173. Abraham, A. & Jain, R. Soft Computing Models for Network Intrusion Detection Systems. *Studies in Computational Intelligence* **16**, 191–211 (2005). (Cited on pages 142 and 189.)

174. Gharibian, F. & Ghorbani. A. A. A Comparative Study of Supervised Machine Learning Techniques for Intrusion Detection. In *Proceedings of the Fifth Annual Conference on Communication Networks and Services Research(CNSR'07)*, 350–358 (IEEE Computer Society Press, Fredericton, New Brunswick, Canada, 2007). (Cited on page 142.)

175. Stein, G., et al. Decision tree classifier for network intrusion detection, with GA-based feature selection. In *ACM-SE 43 Proceedings of the 43rd Annual Southeast Regional Conference*, 136–141, (2005). (Cited on page 142.)

176. Zhang, J. & Zulkernine, M. Network intrusion detection using random forest. In *Proceedings of the Third Annual Conference on Privacy, Security and Trust*, 53–61, (2005). (Cited on page 142.)

177. Li, Y., et al. TCM-KNN Algorithm for Supervised Network Intrusion Detection. *Lecture Notes in Computer Science* **4430/2007**, 141–151 (2007). (Cited on pages 142 and 143.)

178. Gogoi, P., et al. Anomaly Detection Analysis of Intrusion Data using Supervised & Unsupervised Approach. *Journal of Convergence Information Technology* **5**(1), 95–110, Feb. (2010). (Cited on pages 142 and 207.)

179. Horng, S.-J., et al. A novel intrusion detection system based on hierarchical clustering and support vector machines. *Expert Systems with Applications* **38**, 306–313 (2011). (Cited on pages 142, 143, 180, 231 and 256.)

180. Borah, B. & Bhattacharyya, D. K. CatSub: A Technique for Clustering Categorical Data Based on Subspace. *The ICFAI University Journal of Computer Sciences* **II**(2), 7–20, April (2008). (Cited on page 144.)

181. Veltkamp, R. C. & Latecki, L. J. Properties and Performances of Shape Similarity Measures. In Content-Based Retrieval, Crawford, T. and Veltkamp, R. C., editors, number 06171 in Dagstuhl Seminar Proceedings (Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, Dagstuhl, Germany, 2006). (Cited on page 146.)

182. Mukkamala, S. & Sung, A. H. Significant Feature Selection Using Computational Intelligent Techniques for Intrusion Detection. In *Advanced Methods for Knowledge Discovery from Complex Data*, 285–308 (Springer, London, 2005). (Cited on page 155.)

183. Kayacik, H. G., et al. Selecting Features for Intrusion Detection: A Feature Relevance Analysis on KDD 99 Intrusion Detection Datasets. In *Proceedings of the 3rd Annual Conference on Privacy, Security and Trust* (Dalhousie University, Halifax, NS, Canada, 2005). (Cited on pages 155 and 156.)

184. Jesen, F. V. Introduction to Bayesien networks. UCL Press, (1996). (Cited on pages 180 and 256.)

185. Jordan, M. I. & Bishop, C. M. Neural Networks. In *Allen B Tucker Computer Science Handbook* (Chapman & Hall/CRC Press LLC, Boca Raton, 2004). (Cited on page 184.)

186. Jain, A. K., et al. Data Clustering: A Review. *ACM Computing Survey* **31**(3), 264–323 (1999). (Cited on page 185.)

187. Otey, M. E., et al. Fast distributed outlier detection in mixed-attribute data sets. *Data Mining and Knowledge Discovery* **12**(2-3), 203–228 (2006). (Cited on page 188.)

188. Bay, S. D. & Schwabacher, M. Mining distance-based outliers in near linear time with randomization and a simple pruning rule. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 29–38. ACM Press, (2003). (Cited on page 188.)

189. Ramaswamy, S., et al. Efficient algorithms for mining outliers from large data sets. *ACM SIGMOD Record* **29**(2), 427–438, June (2000). (Cited on page 189.)

190. Noh, S., et al. Compiling Network Traffic into Rules Using Soft Computing Methods for the Detection of Flooding Attacks. *Applied Soft Computing* **8**(3), 1200–1210 (2008). (Cited on page 189.)

191. Toosi, A. N. & Kahani, M. 'A New Approach to Intrusion Detection Based on an Evolutionary Soft Computing Model using Neuro-Fuzzy Classifiers. *Computer Communications* **30**, 2201–2212 (2007). (Cited on page 189.)

192. Song, D., et al. Training Genetic Programming on Half a Million Patterns: An Example from anomaly Detection. *IEEE Transactions on Evolutionary Computation* **9**(3), 225–239 (2005). (Cited on page 189.)

193. Abadeh, M. S., et al. Intrusion detection using a fuzzy genetics-based learning algorithm. *Journal of Network and Computer Applications* (2005). (Cited on page 189.)

194. Gomez, J. & Dasgupta, D. Evolving Fuzzy Classifiers for Intrusion Detection. In *Proceeding of 2002 IEEE Workshop on Information Assurance*, 68–75 (United States Military Academy, West Point, NY, USA, 2001). (Cited on page 189.)

195. Zhang, R., et al. Network Anomaly Detection Using One Class Support Vector Machine. In *Proceedings of the International MultiConference of Engineers and Computer Scientists Vol I (IMECS 2008)*, 452–456 (Newswood Limited, Hong Kong, 2008). (Cited on page 190.)

196. Oldmeadow, J., et al. Adaptive clustering for network intrusion detection. In *Proceedings of the Third International Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2004)*, volume 3056 of *LNCS*, 255–259 (Springer, Sydney, Australia, 2004). (Cited on pages 190 and 203.)

197. Eskin, E., et al. A Geometric Framework for Unsupervised Anomaly Detection: Detecting Intrusions in Unlabeled Data. In *Applications of Data Mining in Computer Security*, 78–100 (Kluwer Academic Publishers, Norwell, MA, USA, 2002). (Cited on pages 190, 203 and 232.)

198. Nagesh, H. S., et al. A scalable parallel subspace clustering algorithm for massive data sets. In *Proceedings of the ICPP 2000*, 477 (IEEE Computer Society, Toronto, Canada, 2000). (Cited on page 190.)

199. Zhang, J. & Zulkernine, M. Anomaly based network intrusion detection with unsupervised outlier detection. In *IEEE International Conference on Communications (ICC)*, 2388–2393 (IEEE Xplore, Istanbul, 2006). (Cited on page 191.)

200. Breiman, L. Random Forests. *Machine Learning* **45**(1), 5–32 (2001). (Cited on page 191.)

201. Mitchell, T. M. *Machine Learning*. McGraw-Hill, Inc., New York, USA, (1997). (Cited on page 206.)

202. Paxson, V. Bro: A System for Detecting Network Intruders in Real-Time. *Computer Networks* **31**(23-24), 2435–2463 (1999). (Cited on page 207.)

203. Padmanabhan, J. & Easwarakumar, K. S. Traffic engineering based attack detection in active networks. *Lecture Notes in Computer Science (LNCS)* **5408**, 181–186 (2009). (Cited on page 207.)

204. Lazarevic, A., et al. A Comparative Study of Supervised Machine Learning Techniques for Intrusion Detection. In *Proceedings of the Third SIAM International Conference on Data Mining*, 25–36 (SIAM, San Francisco, CA, 2003). (Cited on page 207.)

205. Petrovskiy, M. I. Outlier detection algorithms in data mining systems. *Programming and Computer Software* **29**(4), 228–237 (2003). (Cited on page 208.)

206. Yang, P. & Huang, B. Pacific-Asia Workshop on Computational Intelligence and Industrial Application. In *Proceedings of Pacific-Asia Workshop on Computational Intelligence and Industrial Application*, 511–514. IEEE Computer Society, (2008). (Cited on page 209.)

207. Jin, W., et al. Ranking outliers using symmetric neighborhood relationship. In *Proceedings of the 10th Pacific-Asia conference on Advances in Knowledge Discovery and Data Mining*, 577–593 (Springer-Verlag Berlin, Heidelberg, Singapore, 2006). (Cited on pages 209, 213, 226, 234 and 243.)

208. Jiangab, F., et al. A rough set approach to outlier detection. *International Journal of General Systems* **37**(5), 519–536, Oct. (2008). (Cited on pages 210 and 211.)

209. Hawkins, S., et al. Outlier detection using replicator neural networks. In *Proceedings of the 4th International Conference on Data Warehousing and Knowledge Discovery*, 170–180 (Springer-Verlag, London, UK, 2002). (Cited on page 211.)

210. Song, X., et al. Conditional Anomaly Detection. *IEEE Transactions on Knowledge and Data Engineering* **19**(5), 631–645 (2007). (Cited on page 211.)

211. Tao, Y., et al. Efficient and accurate nearest neighbor and closest pair search in high dimensional space. *ACM Transactions on Database Systems (TODS)* **35**(3), 20:1-20:46, July (2010). (Cited on page 213.)

212. Blake, C. L. & Merz, C. J. UCI Machine Learning Repository. http://www.ics.uci. edu. /~mlearn/MLRepository.html, (2001). (Cited on pages 213, 219 and 220.)

213. Breunig, M. M., et al. LOF: Identifying Density-Based Local Outliers. *ACM SIGMOD* **29**(2), 93–104, June (2000). (Cited on pages 221, 222, 225, 226 and 227.)

214. Bay, S. D. & Schwabacher, M. Mining distance-based outliers in near linear time with randomization and a simple pruning rule. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 29–38 (ACM New York, Washington, DC, USA, 2003). (Cited on pages 221, 222, 225, 226 and 227.)

215. Elkan, C. Results of the KDD'99. http://www.cs.used.edu/users/elkan/clresults.html., (1999). (Cited on page 225.)

216. Zhiyi, Q. & Wenxiu, Z. Anomaly Detection Based on Symmetric Neighborhood Relationship. In *Proceedings of 2nd International Conference on Innovative Computing, Information and Control, 2007, (ICICIC'07)*, 583–587 (IEEE Computer Society Washington, DC, USA, Kumamoto, 2007). (Cited on page 225.)

217. Bace, R. & Mell, P. Intrusion detection systems. NIST Special Publications SP 800, U S Department of Defence, 31 November 2001, (2001). (Cited on page 230.)

218. Last, M., et al. Improving Stability of Decision Trees. *International Journal of Pattern Recognition and Artificial Intelligence (IJPRAI)* **16**(2), 145–159 (2002). (Cited on page 231.)

219. Hastie, T., et al. *The Elements of Statistical Learning- Data Mining, Inference, and Predicting, 2nd Edition*. Springer, Stanford, California, (August 2008). (Cited on page 231.)

220. Aha, D. W., et al. Instance-Based Learning Algorithms. *Machine Learning* **6**(1), 37–66 (1991). (Cited on page 231.)

221. Garcia-Pedrajas, N. & Ortiz-Boyer, D. Boosting k-nearest neighbor classifier by means of input space projection. *Expert Systems with Applications* **36**(7), 10570–10582 (2009). (Cited on page 231.)

222. Pan, Z. S., et al. Hybrid neural network and C4.5 for misuse detection. In *Proceeding of International Conference on Machine Learning and Cybernetics*, volume 4, 2463–2467, (2003). (Cited on page 234.)

223. Cheeseman, P., et al. AutoClass: A Bayesian classification system. In *Proceedıngs of the Fifth Internatıonal Conference on Machıne Learnıng (ML'88)*, volume 27, 54–64. Morgan Kaufmann, (1988). (Cited on page 236.)

224. Huberi, L. & Schultz, J. Quadratic assignment as a general data analysis strategy. *Brıtısh Journal of Mathematıcal and Statıstıcal Psychology* **29**(2), 190–241 (1976). (Cited on pages 240 and 245.)

225. Gogoi, P., et al. Supervised Anomaly Detection using Clustering-based Normal Behaviour Modeling. *Internatıonal Journal of Advances ın Engıneerıng Scıences* **1**(1), 12–17 (2011). (Cited on pages 243 and 260.)

226 Gogoi, P., et al Network Anomaly Detection Using Unsupervısed Model *Internatıonal Journal of Computer Applıcatıons (Specıal Issue on Network Security and Cryptography)* **NSC**(1), 19–30, Dec. (2011). (Cited on pages 243 and 260.)

227. Guo, Y , et al. Feature selection based on rough set and modified genetic algorithm for intrusion detection. In *In Proceeding of 5th Internatıonal Conference on Computer Science & Educatıon*, 1441–1446 (IEEEXplore, Hefei, China, 2010). (Cited on page 246.)

228. University of Waikato. Weka 3: Data Mining Software in Java. http://www.cs.waikato.ac.nz/ml/weka, (1999). (Cited on page 256.)

229. Gogoı, P., et al. Outlıer Identification using Symmetric Neighborhoods. In *Proceedıngs of 2nd Internatıonal Conference on Communıcatıon Computing & Securıty (ICCCS 2012.* (Cited on page 260.)

230. Peng. T., et al. Proactively detecting distributed denial of service attacks using source ip address monitoring. In *Proceedıngs of the 3rd Internatıonal IFIP-TC6 Networking Conference (Networkıng 2004)*, 771–782, (2004). (Cited on page 262)

Author's Publications

1. Prasanta Gogoi, B. Borah and D. K. Bhattacharyya, Anomaly Detection Analysis of Intrusion Data using Supervised & Unsupervised Approach, Journal of Convergence Information Technology, Volume 5, Number 1, February 2010, pp 95-110, citation: 20.

2. Prasanta Gogoi, D.K. Bhattacharyya, B. Borah and J. K. Kalita, A Survey of Outlier Detection Methods in Network Anomaly Identification, The Computer Journal, Oxford University Press, Volume 54, Number 4, April 2011, pp 570-588, citation: 14.

3. Prasanta Gogoi, B. Borah and D. K. Bhattacharyya , Supervised Anomaly Detection using Clustering-based Normal Behavior Modeling, International Journal of Advances in Engineering Sciences, Vol. 1(1), pp 12-17, 2011 e-ISSN: 2231-0347 print-ISSN: 2231-2013.

4. Prasanta Gogoi and B. Borah and D. K. Bhattacharyya, 'Network Anomaly Detection Using Unsupervised Model', International Journal of Computer Applications (Special Issue on Network Security and Cryptography) , Vol. NSC(1), pp 19-30, Foundation of Computer Science, New York, USA, December, 2011 [DOI:10.5120/4321-008].

5. Prasanta Gogoi, R. Das, B. Borah, and D. K. Bhattacharyya ,'Efficient Rule Set Generation using Rough Set Theory for Classification of High Dimensional Data', International Journal of Smart Sensors and Ad Hoc Networks (IJSSAN), Vol. 1(2), pp 13-20, 2011 [ISSN No. 2248-9738].

6. Prasanta Gogoi, M H Bhuyan, D K Bhattacharyya and J K Kalita, Packet and Flow Based Network Intrusion Dataset, In Proceedings of the 5th International Conference on Contemporary Computing (IC3-2012), August 6-8, India, pp 322-334, CCIS 306, Springer, 2012 [ISSN: 1865-0929].

7. Prasanta Gogoi, B Borah and D K Bhattacharyya, Outlier Identification using Symmetric Neighborhood, in Proceedings of the 2nd International Conference on Communication, Computing & Security, (ICCCS 2012), October 6-8, India, the Elsevier Procedia Technology, vol. 6, pp 239-246, Elsevier, 2012. [ISSN: 2212-0173].

8. Prasanta Gogoi, B. Borah and D. K. Bhattacharyya, 'Classification of Network Anomalies using Clustering', Informatica, an International Journal of Computing and Informatics, Volume 37, Number 1, 2013, pp 93-105, the Slovenian Society Informatika, [print ISSN: 0350-5596, web ISSN: 1854-3871].

9. Prasanta Gogoi, D. K. Bhattacharyya and J. K. Kalita,'A Roughset Based Effective Rule Generation Method for Classification with an Application in Intrusion Detection', International Journal of Security and Networks (IJSN), Inderscience (in Press) [Online ISSN: 1747-8413, Print ISSN: 1747-8405].

10. Prasanta Gogoi, B. Borah and D. K. Bhattacharyya,'MLH-IDS: A Multi-level Hybrid Intrusion Detection Method', The Computer Journal, Oxford University Press, (in Press) [Online ISSN: 1460-2067, Print ISSN: 0010-4620].