Applying Data Mining Techniques in Anomaly Based Network Intrusion Detection

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

Monowar Hussain Bhuyan

Registration No. 045 of 2011

School of Engineering

Department of Computer Science and Engineering

Tezpur University

September 2013

Dedication

This thesis is dedicated to my beloved son Zeeshan and wife, Rehena without whose love support and inspiration I would never have made it this far I also wish to dedicate this thesis to my parents who have supported me all the way since the beginning of my studies

To my beloved Son, Wife, Father, Mother and Sisters

Abstract

With the enormous growth in network connectivity resulting in high bandwidth Internet services, and huge increase in the number of Internet-based applications, network security is becoming increasingly important Almost all computer systems suffer from security vulnerabilities which are economically costly and impossible to be solved by manufacturers alone Therefore the role of intrusion detection systems as special purpose applications to detect anomalies and attacks in a network is becoming very important Traditional intrusion detection systems are reactive in the sense that they use a set of signatures to identify malicious traffic patterns, the size of the signature set grows as new vulnerabilities are discovered Anomaly detection systems are a category of intrusion detection systems that act more proactively They derive a model of the normal system behavior and issue alarms whenever the behavior changes, making a stable assumption that such changes are frequently caused by malicious or disruptive events Anomaly detection has been a field of intense research over the years as it poses many challenging problems Data mining techniques have proven to be useful in effective identification of anomalous traffic patterns

The contributions of this thesis pertain to the area of network traffic anomaly detection using data mining techniques This thesis has three parts including background and literature review a systematic approach to generate real-life network intrusion datasets, and approaches for network anomaly detection In the first part, it presents an overview of networks attack a taxonomy and categories of detection methods with architectures, and an extensive review of network anomaly detection methods, systems and tools In the second part, it introduces a systematic approach to generate real-life network intrusion datasets using the TUIDS testbed architecture such up-to-date datasets are useful for the network security research community to test detection methods and systems In the third part, this thesis introduces approaches for network anomaly detection It presents an outlier-based approach by introducing an outlier score function to rank each candidate object and report traffic patterns as normal or coordinated scan as early as possible This thesis includes a tree-based subspace clustering technique for high dimensional large datasets It generates reference points to estimate outlier score

values to detect large scale network anomalies. This thesis also includes an unsupervised approach for network anomaly detection in large datasets with a focus on detection of known as well as unknown attacks without using any labelled traffic or signatures or training. The approach includes two algorithms, viz., TreeCLUSS and CLUSSLab. CLUSSLab is an effective cluster labelling technique to label each cluster based on a stable cluster set obtained from TreeCLUSS using multiple objectives. It also presents an effective unsupervised feature clustering technique to identify a dominant feature subset for each cluster, to be used for cluster labelling. Finally, this thesis includes an extended entropy metric-based DDoS flooding attack detection approach to detect four classes DDoS attacks, viz., constant rate, pulsing rate, increasing rate and subgroup attacks. It can successfully identify the DDoS attacks by measuring the metric difference between legitimate traffic and attack traffic using extended entropy metric. This approach also extends the mechanism to use an ensemble of extended entropy metrics for increasing detection rate in near real-time. The proposed techniques are validated using real-life datasets and have been found to perform well in comparison to competing algorithms. All network anomaly detection algorithms have been validated in terms of detection rate, false positive rate, ROC, precision, recall and F-measure.

Keywords — Coordinated Scan, Outlier Detection, Port Scan, Anomaly Detection, DoS, Score, Cluster, Attack, Reference Point, Profile, DDoS, Intrusion Detection, Entropy Metric

Tezpur University

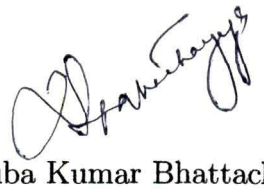
Certificate

This is to certify that the thesis entitled *Applying Data Mining Techniques in Anomaly-Based Network Intrusion Detection* submitted to the Tezpur University in the Department of Computer Science and Engineering under the School of Engineering in partial 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. Monowar Hussain Bhuyan under my personal 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.

Signature of Research Supervisor



(Dhruba Kumar Bhattacharyya)
Designation: Professor and Dean
School: Engineering
Department: Computer Science and Engineering

University of Colorado at Colorado Springs

Department of Computer Science
1420 Austin Bluffs Parkway
P.O. Box 7150
Colorado Springs, Colorado 80933-7150
(719) 255-3325
(719) 255-3369 Fax

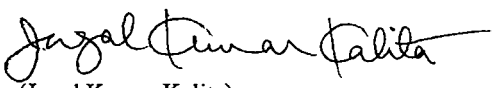
September 3, 2013

Certificate

This is to certify that the thesis entitled *Applying Data Mining Techniques in Anomaly-Based Network Intrusion Detection* submitted to the Tezpur University in the Department of Computer Science and Engineering under the School of Engineering in partial 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. Monowar Hussain Bhuyan under my personal supervision and guidance.

All help received by him from various sources has been duly acknowledged.

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



(Jugal Kumar Kalita)
Designation: Professor
College: Engineering and Applied Science
Department: Computer Science

Declaration

I, Monowar Hussain Bhuyan, hereby declare that the thesis entitled *Applying Data Mining Techniques in Anomaly-Based Network Intrusion Detection* submitted to the Department of Computer Science and Engineering under the School of Engineering, Tezpur University, in partial fulfillment of the requirements for the award of the degree of Doctor of Philosophy is based on bonafide work carried out by me. The results embodied in this thesis have not been submitted in part or in full, to any other university or institute for award of any degree or diploma.

Monowar Hussain Bhuyan
16.09.2012
(Monowar Hussain Bhuyan)

Acknowledgements

This Thesis would not have been possible without the support of a lot of people whose help has been fundamental during the years of my Ph D studies

I would like to express my deepest and sincere appreciation to Professor Dhruba Kumar Bhattacharyya my research supervisor for his worthwhile assistance, guidance, advice support and endless patience throughout the course of this research I would like to thank him for his continuous efforts to show me the path to follow in each and every steps This Thesis represents my commitment to shape his brilliant and creative scientific ideas in applying data mining techniques to solve network security problems It was a honor for me to work with him

I would like to extend my heartfelt gratitude to my co-supervisor Professor Jugal Kumar Kalita whose constant encouragement and valuable insights provided the perfect guidance that I needed through those years of research He is a wonderful advisor, mentor, and motivator Throughout my research he furnished me countless constructive comments and suggestions that make my thesis flawless

I am also very thankful to the rest of my thesis guidance committee including Professor Smriti Kumar Sinha and Dr Bhogeswar Borah Their advice and suggestions have been very helpful and useful

I owe my gratitude to all my colleagues of the Ph D School who gladden these years of studies in Tezpur, and to all my friends who supported me also during those periods in which I was too taken from my work to reciprocate

My deep gratitude goes to my beloved son and my wife I can never express my thanks enough for their endless love support and understanding

Towards the end, gratitude goes to my loved mother and father back home for their endless love and sacrifices, and for their sustained moral and financial support during those years of research and throughout my whole life You were always my source of strength

This research was supported in part by the Department of Information Technology (DeitY) and Council of Scientific and Industrial Research (CSIR) Government of India The author gratefully acknowledges their financial support

Tezpur University

Certificate

This is to certify that the thesis entitled *Applying Data Mining Techniques in Anomaly-Based Network Intrusion Detection* submitted by Mr. Monowar Hussain Bhuyan to Tezpur University in the Department of Computer Science and Engineering under the School of Engineering in partial 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

Principal Supervisor

External Examiner

Date:_____

Contents

List of Tables

List of Figures

List of Symbols

I	Intrusion detection system	24
M	Model of normal behavior	24
D	Proximity measure	24
A	Anomaly score	31
$\mathbb{X}$	Dataset	142
FC	Feature correlation	116
$\mathbb{F}$	Total traffic features	116
$\mathbb{N}$	Total traffic instances	116
α_1	Basic features	116
β_1	Content-based features	116
γ_1	Time-based features	116
δ_1	Connection-based features	116
$\mathbb{T}$	Time window taken for processing	116
F'	Relevant feature subset	143
δ_2	Threshold for coordinated scan detection	142
n	Number of data object in X	144
r	Real number greater than 1	146
l_1	Number of iterations	146
ϕ	Termination criteria between 0 and 1	146
dist	Dissimilarity measure between two object O_i and O_j	148
d	Total number of dimensions in X	167
x_i	i^{th} data object	167
$X_{c/t}$	Candidate or test data objects	176
$\mathbb{C}$	Set of clusters	171
R_i	i^{th} reference point	176
S_i	Occurrence belonging to a class within k' nearest neighbor	176

C_i	i^{th} group or set of outlier instances	165
C_i^N	i^{th} group or set of normal instances	165
τ	Threshold value for outlier score	177
μ	Mean-based profile value with respect to a cluster	149
k'	Number of nearest neighbor	147
m	Cardinality of optimal subset of features	176
m'	Number of large clusters	176
k	Number of clusters	176
ξ	Threshold for feature selection	171
α'	Proximity threshold	172
β'	Minimum number of data objects in the neighborhood of an initiator	172
l	Level of the tree	172
$n_{F'}$	Total number of relevant features	173
min	Minimum rank value found with respect to IGFS algorithm	173
α''	Random subset selection using maximum length pseudo random sequence generator	149
θ	Height of the tree	173
CL	Class label	174
N_i	i^{th} node in the tree	208
P'	Matching probability of dominant feature subset	217
α_2	Threshold for L_1 cluster	208
β_2	Threshold for L_2 cluster	208
σ_1	Minimum threshold value for cluster stability checking	211
σ_2	Maximum threshold value for cluster stability checking	211
λ	Renyi s constant	214
σ	Smoothing parameter in Gaussian Kernel	215
η_1	Threshold value for inter-cluster entropy	216
η_2	Threshold value for intra-cluster entropy	216
ξ_1	Number of instances in a cluster	217
ξ_2	Cluster compactness score	217
ξ_3	Threshold for matching probability of features of a cluster with respect to a specific class	217
ξ_4	Outlier score value of each instance of a cluster	217

List of Abbreviations

DoS	Denial of Service	1
KDD	Knowledge Discovery in Databases	2
DMZ	Demilitarized Zone	3
HTTP	Hypertext Transfer Protocol	4
FTP	File Transfer Protocol	4
SMTP	Simple Mail Transfer Protocol	4
ITU	International Telecommunication Union	4
IDS	Intrusion Detection System	7
ANIDS	Anomaly based Network Intrusion Detection System	7
DDoS	Distributed Denial of Service	10
TUIDS	Tezpur University Intrusion Detection System	11
SQL	Structured Query Language	19
TCP	Transmission Control Protocol	19
IMAP	Internet Message Access Protocol	19
POP	Post Office Protocol	19
IP	Internet Protocol	19
DNS	Domain Name System	20
HIDS	Host-based Intrusion Detection System	21
NIDS	Network-based Intrusion Detection System	22
BPF	Berkeley Packet Filter	24
UDP	User Datagram Protocol	27
ICMP	Internet Control Message Protocol	27
MINDS	Minnesota INtrusion Detection System	27
R2L	Remote to Local	28
U2R	User to Root	28
RMHC	Random Mutation Hill Climbing	38

z	Total dominating feature subset obtained from each cluster	217
γ_2	Threshold for class specific feature selection	209
sim	Proximity between two objects O_i and O_j	209
ε	A factor for step down ratio	209
$\mathbb{P}$	Total probability	255
t_i	i^{th} time interval within T	256
H	Represent entropy metric	255
α	Entropy metric of order α	255
f_i	i^{th} sample packet intensity	256
π	Number of packets	256
S	Sample	259
E	Difference of extended entropy metric values between two samples, S_i and S_j	259
ω_1	Threshold for within sample entropy metric value	252
ω_2	Threshold for relative sample entropy metric value	252
ω_3	Threshold for ensemble-based detection scheme	259

SVM	Support Vector Machine	38
ROC	Receiver Operating Characteristics	42
TPR	True Positive Rate	41
FPR	False Positive Rate	41
TNR	True Negative Rate	41
FNR	False Negative Rate	41
CAIDA	Cooperative Association for Internet Data Analysis	90
LBNL	Lawrence Berkeley National Laboratory	90
LBNL	Defense Advanced Research Projects Agency	90
LLDOS	Lincoln Laboratory DoS	97
CTF	Capture The Flag	97
SSH	Secure Shell	101
VLAN	Virtual Local Area Network	102
LOIC	Low Orbit Ion Cannon	104
TFN2K	Tribe Flood Network 2000	104
NMAP	Network Mapper	104
RNMAP	Remote Network Mapper	105
IRC	Internet Relay Chat	106
RRD	Round Robin Database	108
UCIML	UCI Machine Learning Repository	159
FreeBSD	Berkeley Software Distribution	125
NSM	Network Security Monitor	127
FSD	Flow Size Distribution	132
PHAD	Packet Header Anomaly Detection	133
ROS	Reference-based Outlier Score	144
PCA	Principal Components Analysis	144
DHCP	Domain Host Configuration Protocol	150
TANN	Triangle Area Nearest Neighbors	159
CBUID	Clustering-based Unsupervised Intrusion Detection	159
LOF	Local Outlier Factor	161
IGFS	Information Gain-based Feature Selection	169
CFS	Correlation-based Feature Selection	169
PAM	Partitioning Around Medoids	173

CLARA	Clustering Large Applications	173
CART	Classification and Regression Tree	190
ID3	Iterative Dichotomiser 3	196
HC-SVM	Hierarchical Clustering SVM	196
MMIFS	Modified Mutual Information-based Feature Selection	206
DFS	Dominating Feature Subset	207
TTL	Time to Live	237
DCP	Distributed Change Point	242
CAT	Change Aggregation Tree	242
ISP	Internet Service Provider	243
KNN	K-Nearest Neighbor	245
EEM	Extended Entropy Metric	258

Chapter 1

Introduction

With advancements in network technologies, Internet services have been undergoing constant growth in network traffic, accompanied by an increasing number of anomalies such as Denial of Service (DoS) attacks, port scans, worms, virus exploits and misconfigurations. These anomalies represent a large fraction of the Internet traffic that is unwanted and prevent legitimate users from accessing network resources in an optimal manner. Therefore, detecting and diagnosing these threats are crucial tasks for network operators to ensure that the Internet resources remain available. Because legitimate traffic must be able to travel efficiently, quickly and accurately identifying anomalies in network traffic is important, requires development of good detection techniques. Anomalies are patterns of interest to network defenders, who want to extract them from vast amount of network traffic data. Data mining techniques have been popular in extracting these harmful patterns from large volumes of data in recent years Data mining is used in many application areas, e.g., the business world, medicinal sciences, physical sciences and engineering to make new discoveries. Extensive studies have been performed in applying data mining techniques to network traffic anomaly detection, but the methods [11–13] have limitations that notably discredit them from use in real environments. In this thesis, we explore the possibilities of applying data mining techniques in identifying network anomalies with significant performance improvement.

1.1 Data Mining

Data mining has gained a great deal of attention in the information industry and in the society as a whole in recent years due to the wide availability of huge amounts of data and the imminent need for turning such data into useful information and knowledge. Based on Tan et al [14, 15], data mining is defined as follows

Definition 1.1.1. *Data mining is the process of automatically discovering useful information in large repositories Data mining techniques are deployed to scour large databases in order to find novel and useful patterns that might otherwise remain unknown*

The term knowledge discovery in databases (KDD) refers to the process of converting raw data into useful information or knowledge. Data mining is a step in the KDD process, and applies a variety algorithms for extracting patterns from data. In addition to this, the KDD process has additional steps including data preparation, data selection, data cleaning, incorporation of appropriate prior knowledge, and proper interpretation of the results of mining to ensure useful knowledge is derived from the data

1.1.1 Data Mining Tasks

Based on how they work, data mining tasks are classified into different classes In general, data mining tasks are partitioned into two major categories: *predictive* and *descriptive*. Predictive mining performs inference on the current data in order to make future predictions Descriptive mining characterizes the general properties of the data and underlying relationships among them Some of the most important data mining tasks [14, 15] are discussed below.

(a) *Classification and Regression*: Classification is the process of classifying a data instance into one of several predefined categorical classes based on the training set containing known observations. A regression task begins with data instances in which the target values are known The relationships between predictors and the target are summarized in a regression model that can be applied to different data instances in which the target values are unknown.

(b) *Cluster Analysis*: Cluster analysis seeks to find groups of closely related data objects based on relationships among them. The greater the similarity within a group and the greater the difference between groups, the better is the clustering.

(c) *Association Analysis*: Association analysis is the task of efficiently discovering the most important and the most strongly associated feature patterns in data. The discovered patterns are represented in the form of implication rules.

(d) *Evolution Analysis*: Data evolution analysis describes and models regularities or trends in objects whose behavior changes over time.

(e) *Outlier Detection*: Outlier detection refers to recognizing those observations whose characteristics are significantly different from the rest of the data. These observations are known as outliers.

We concentrate on three major tasks in data mining viz., clustering, classification and outlier mining for identification of network traffic anomalies. Recently, there has been a realization that data mining can have significant impact on network security, especially network traffic analysis. Because most security systems are developed based on the interestingness of traffic patterns, it is important to discover interesting patterns correctly from large datasets as well as analyze them using effective data mining techniques. Data mining can detect known as well as unknown attacks.

1.2 Modern Networks and The Internet

A network is a group of systems that are connected to allow sharing of resources including files, printers or storage media, and sharing of services including basic Internet connection. There are two main aspects to setting up a network: (a) the hardware used to connect the systems together and (b) the software installed on the hosts to allow them to communicate [16]. A typical view of a large network with a demilitarized zone (DMZ)[1] is given in Figure 1.1.

[1]Demilitarized zone is a network segment located between a secured local network and unsecured external networks (Internet). DMZ usually contains servers that provide services to users on the external network, such as web, mail, and DNS servers, which must be hardened systems. Two firewalls are typically installed to form the DMZ.

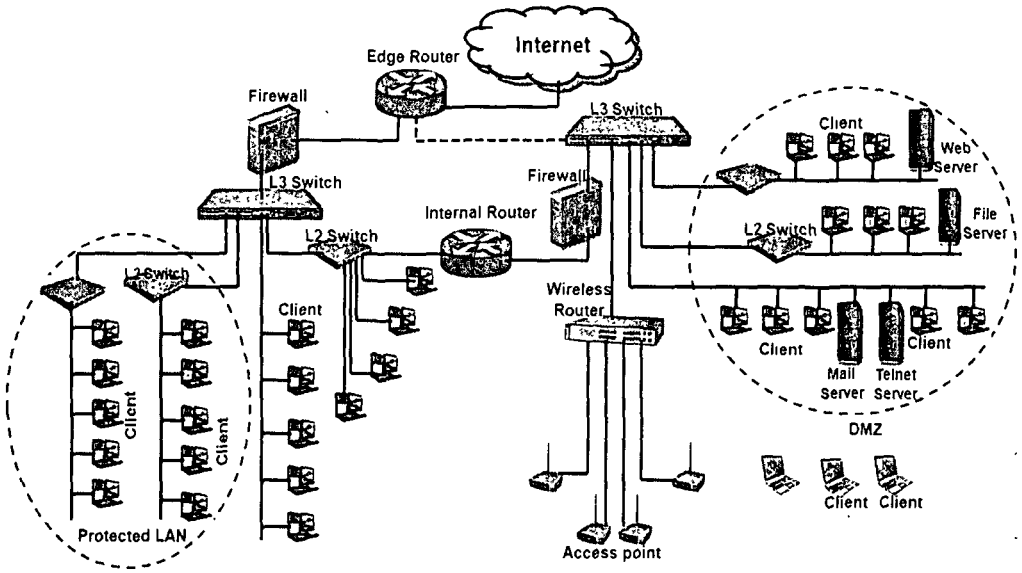


Figure 1.1: A typical view of an enterprise network with DMZ

A typical network involves several hosts connected through network devices and users to share data and resources with each other. A device or system that is connected to the network is known as a host. The workstation is a general purpose host with high end configuration for technical and scientific applications. The sever is a special host that contains more disk space and memory than are found on clients (i.e., hosts, workstations). A sever has special software installed that allows it provide the intended function. It provides services such as file and print services, serving Web pages to clients, controlling remote access and security to clients. The *Internet* is the world wide network of computers accessible to anyone through protocols such as HTTP, FTP or SMTP. Day by day we have become increasingly dependent on the Internet as users. We present statistics of the worlds Internet users in Figure 1.2, as reported by the International Telecommunication Union (ITU)[1].

Network *vulnerabilities* are the inherent weaknesses in the design, configuration, or implementation of a computer network that renders it susceptible to a security threat. The growth of network vulnerabilities as reported in [17] is shown in Figure 1.3. Threats may arise from exploitation of design flaws in the hardware and software of computer network systems [18]. Systems may also be incorrectly config-

[1]http://www.itu.int/en/Pages/default.aspx

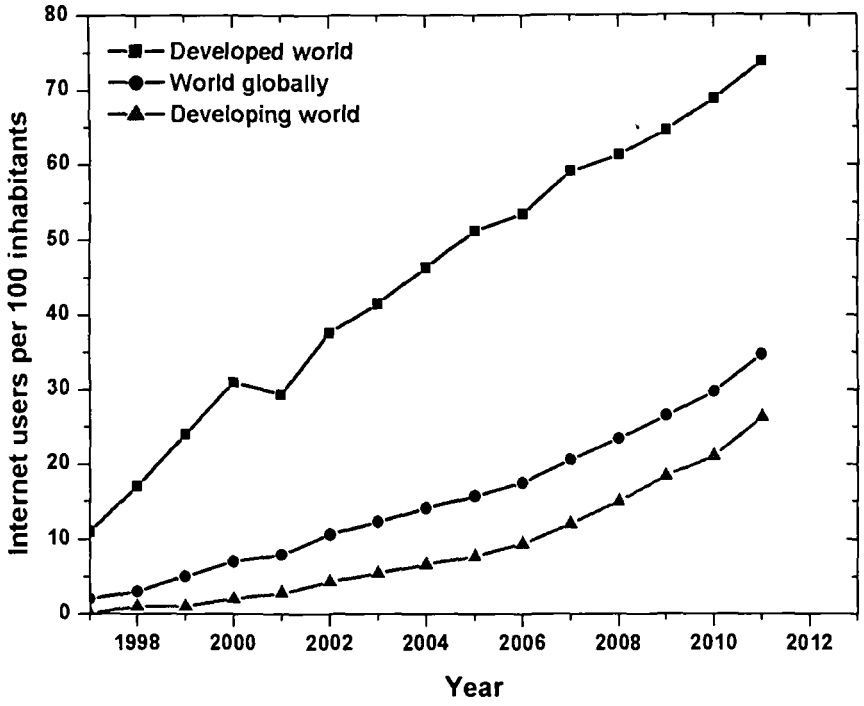


Figure 1.2: The growth of world's Internet users

ured, and therefore vulnerable to attack. The vulnerabilities of this kind generally occur from inexperience, insufficient training, or half done work. Another source of vulnerability is poor management such as inadequate procedures and insufficient checks of the network systems.

1.2.1 Network Anomalies and Detection

Anomalies are instances in data that do not conform the normal behavior. The instances are also known as objects, points, events, vectors, or samples. Anomalies in network can be defined as any network events or operations that deviate from normal network behavior. They happen due to the growing number of network based attacks or intrusions. The recent growth of Internet threat agents[1] is given in Figure 1.4. Network threats may occur due to many reasons including: (i) malicious activities that interpret normal network services, (ii) network overload, (iii) device malfunctioning, and (iv) compromises in different network parameters.

Anomaly detection attempts to find patterns in network traffic data, which

[1] http://www.verizon.com/enterprise/databreach

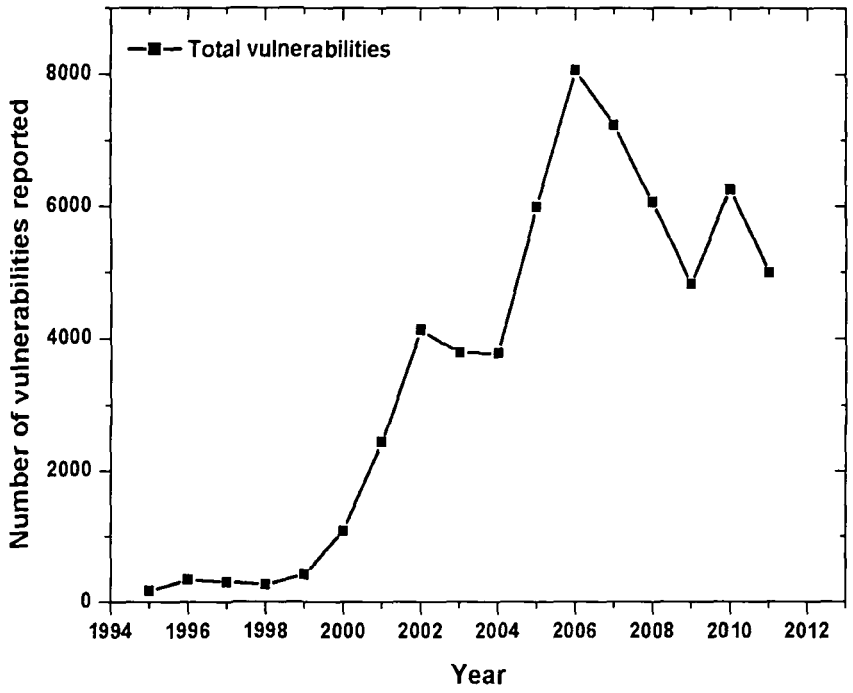


Figure 1.3: The growth of reported network vulnerabilities

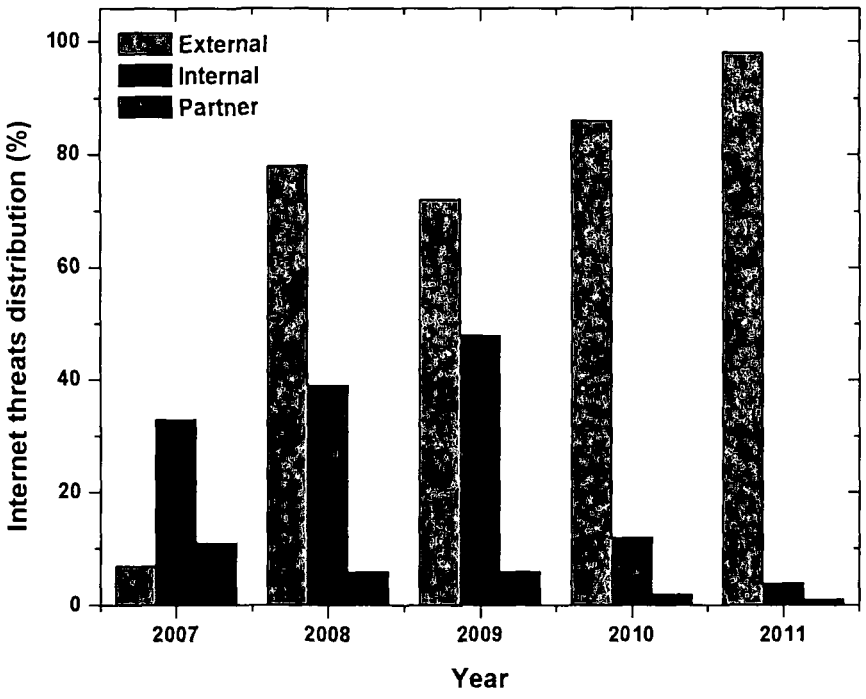


Figure 1.4: The growth of Internet security threats

do not conform to expected normal behavior. The importance of anomaly detection is due to the fact that anomalies in data translate to significant (and often critical) actionable information in a wide variety of application domains [19]. For example, an anomalous traffic pattern in a computer network could mean that a hacked computer is sending out sensitive data to an unauthorized host. Anomalies in a network may be caused by different reasons. As stated in [18], there are two broad categories of network anomalies: (a) performance related anomalies and (b) security related anomalies. Various examples of performance related anomalies are: broadcast storms, transient congestion, babbling node, paging across the network, and file server failure. Security related network anomalies may be due to malicious activities of the intruder(s) by intentional flooding of the network with unnecessary traffic to hijack the bandwidth so that legitimate users are unable to receive service(s). However, our thesis is concerned is with security related network anomalies only. Currently, anomaly based network intrusion detection is the most successful intrusion detection technique. It is currently a principal focus of research and development in the field of intrusion detection. Various systems with ANIDS (Anomaly based Network Intrusion Detection System) capabilities are becoming available, and many new schemes are being explored. However, the subject is far from mature and key issues remain to be solved before wide scale deployment of ANIDS platforms becomes practicable.

Advances in networking technology have enabled us to connect distant corners of the globe through the Internet for sharing information in the vast. However, along with this advancement, threats from spammers, attackers and criminal enterprises are also growing in multiple speed [11]. Normally, an intrusion attempts to compromise the confidentiality, integrity, or availability of a system, or to bypass the security mechanisms of a host or a network. As a result, security experts use intrusion detection technology to keep secure the infrastructure of large enterprises. An intrusion detection System (IDS) is defined [20] as follows.

Definition 1.2.1. *An Intrusion Detection System monitors the events occurring in a computer system or a network and analyzes them for signs of intrusions.*

Intrusion detection systems are divided into two broad categories: *misuse* detection [21] and *anomaly* detection [22] systems. *Misuse* detection can detect only

known attacks based on signatures they have created and stored Thus, dynamic signature updation is important and as a result new attack definitions are frequently released by the IDS vendors. However, misuse based systems cannot cope with the rapidly growing number of vulnerabilities and exploits On the other hand, *anomaly* based detection systems are designed to capture any deviation from the profiles of normal behavior. They are more suitable than misuse detection for detecting unknown or novel attacks without any prior knowledge. But normally such systems generate a large number of false alarms.

There are four commonly used machine learning approaches for detecting intrusions or anomalies in network traffic [12]: (i) supervised, (ii) semi-supervised, (iii) unsupervised and (iv) hybrid. In the *supervised* approach [23–25], a predictive model is developed based on a training dataset that contains normal and attack data instances Any unseen data instance is compared against the model to determine which class it belongs to In the *semi-supervised* approach [26–28], the training data instances contain only the normal class. Data instances are not labeled for the attack class. There are many approaches used to build the model for the class corresponding to normal behavior. This model is used to identify anomalies in the test data. In the *unsupervised* approach [7, 29–31], the model does not require any training data, and thus are potentially most widely applicable. Finally, the *hybrid* approach [32–34] normally exploits the features of all of the above to get effective and efficient performance in detecting network anomalies on a large scale Also, these techniques make the implicit assumption that normal instances are far more frequent than anomalies in the test data. If this assumption is not true, such techniques suffer from high false alarm rates. In the first two cases, it requires training on the instances for finding anomalies But getting a large amount of labelled normal and attack training instances may not be practical for a particular scenario. Again, to generate a set of true normal instances with all the variations is an extremely difficult task. Hence, this thesis develops some effective and excellent network traffic anomaly detection methods to detect known as well as unknown attacks with high detection rate and low false positive rate while compared with competing methods.

1.3 Open Issues in Network Anomaly Detection

Network anomaly detection has become increasingly necessary in recent years due to the proliferation of Internet-based attacks The discovery of promising methods for detecting unknown attacks have made it effective as well It is one of the most important infrastructure security mechanisms for an enterprise network An intrusion is a set of actions aimed to compromise the computer security goals such as confidentiality, integrity and availability of resources Anderson while introducing the concept of intrusion detection [35] in 1980 defined an intrusion attempt or a threat to be the possibility of a deliberate unauthorized attempt to (a) access information, (b) manipulate information, or (c) render a system unreliable or unusable For example (a) *Denial of Service (DoS)* attack attempts to starve a host of its resources, which are needed to function correctly during processing (b) *Worms and viruses* exploit other hosts through the network, and (c) *Compromises* obtain privileged access to a host by taking advantages of known vulnerabilities

Due to the voluminous nature of network traffic data it is important to analyze the data using standard high dimensional data analysing techniques So, data mining approaches are most useful and applicable in this area Most existing Network Intrusion Detection Systems (NIDSs) have been found inadequate in detecting the growing number of novel attacks However anomaly based NIDSs have been somewhat successful in detecting both known as well as unknown attacks in normal systems or in network traffic But they generate high false alarm rates irrespective of any network scenarios Applications of data mining in improving security include intrusion detection, cyber attack detection, criminal detection, biometric authentication, etc Several data mining techniques such as clustering, classification, association analysis and anomaly detection have been useful in discovering relevant knowledge from data sources for detecting novel attacks Hence, we concentrated on applying data mining techniques to network anomaly detection to discover known as well as unknown attacks The following is a list of research issues and objectives for our thesis as we worked towards an effective solution

- A NIDS must be tested and evaluated using real time labelled network traffic traces with a comprehensive and extensive set of intrusion or attack data

before deploying in any real world environment. This is a significant challenge, since availability of such datasets is low. Therefore the generation of a real-life network intrusion dataset is something that remains to be addressed.

- Due to the lack of availability of labeled datasets for training or validation of the models, most attack detection approaches produce high false alarm rates. Thus, minimization of the false alarm rate is a problem that must be tackled.
- With the evolving nature of networking technology and with the constant effort of attackers to launch newer attacks, existing IDSs are non-adaptive and hence inadequate in handling known as well as unknown attacks.
- The voluminous size of network traffic and the constant changes in traffic patterns as well as the presence of noise in the audit data make the task of building profile or signature for normal network traffic a daunting task.
- Network traffic is composed of a large amount of data. If security models generate profiles for normal as well as attack traffic, it is a challenge to update online signature databases dynamically.
- Logic applicable to normal attack identification may not be useful in identifying attacks that are rare.
- Assumptions related to statistical distribution of normal and attack traffic may not be valid in detection of Distributed Denial of Service (DDoS) attacks.

1.4 Contributions

In this thesis, we apply data mining techniques to network traffic anomaly detection. We also evaluate these techniques. In addition, we prepare new network intrusion datasets by incorporating several real world attacks. The main contributions of this thesis are as follows.

- A detailed literature survey has been carried out on applications of data mining techniques to network traffic anomaly detection under six major categories. It includes detection methods, systems and popular tools found in

this domain. We have identified several common pitfalls in network anomaly detection and compared different systems and tools.

- A systematic method for network intrusion dataset preparation is introduced. We have prepared three datasets, viz., (a) TUIDS intrusion dataset (b) Coordinated scan dataset and (c) DDoS flooding attack dataset. These datasets are used for performance evaluation of the proposed methods in addition to the benchmark network intrusion datasets.

- We present an adaptive outlier-based coordinated scan detection approach to detect malicious scans at an early stage. It exploits linear congruential generators to select random normal samples from the whole dataset for training purpose. We introduce an outlier score function to rank each candidate data object with respect to the profiles and report as normal or outlier. The profiles is built using the clusters obtained from fuzzy c-means clustering technique. This method is evaluated using benchmark and real-life datasets.

- A clustering and outlier-based approach for network anomaly detection is proposed. To support the anomaly detection process, we introduce a tree-based clustering algorithm to generate a set of reference points. Finally, it uses our outlier score function to test each candidate object to identify network anomalies using estimated score values. This approach is evaluated using benchmark and real-life datasets.

- An unsupervised approach for network anomaly detection is introduced for high dimensional large network traffic datasets. We introduce an unsupervised tree-based subspace clustering technique with a cluster stability measure. We also propose a cluster labelling technique to label each cluster as anomalous or normal. Benchmark and real-life datasets are used for performance evaluation.

- Finally, we propose an extended entropy metric-based approach for detecting all possible scenarios of DDoS flooding attacks. It estimates extended entropy metric based on Renyi's generalized entropy with packet intensity computation over a sampled network traffic. In addition, we combine the selective order of extended entropy metric to improve the detection accuracy in near

real-time This method is evaluated using benchmark and real-life datasets. In addition, we provide a detailed review of DDoS attacks, taxonomy, detection methods and popular tools.

1.5 Outline of the Thesis

The thesis structure closely follows the research issues identified. Figure 1 5 presents a sketch of the thesis outline, where the chapters have been grouped according to topic and the research contributions. The remainder of this thesis is therefore organized as follows.

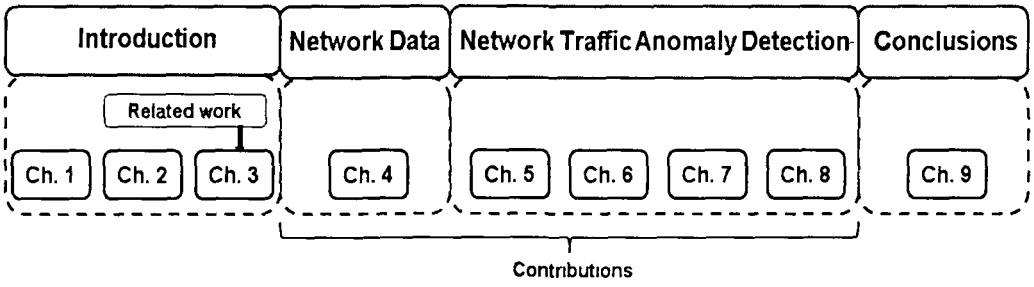


Figure 1.5: Thesis structure

- *Chapter 2* provides the background information needed for the research presented in this thesis. The chapter covers three topics: anomalies in networks, anomaly detection and evaluation of such techniques, and network attacks in the form of a taxonomy.

- *Chapter 3* presents a structured overview progress so far in the field of network anomaly detection. It includes network anomaly detection methods, systems and tools in a taxonomy

- In *Chapter 4*, we discuss several real-life network intrusion datasets that are normally used for evaluation of network anomaly detection methods or systems. We present a systematic method to prepare real-life network intrusion datasets by incorporating several attacks.

- *Chapter 5* initially presents a review on port scan and their detection methods. Then we propose an adaptive outlier-based approach for coordinated scan

detection and also to detect malicious scans at an early stage We introduce an outlier score function to rank each candidate data object with respect to the profiles. The profiles are built using the clusters obtained from fuzzy c-means clustering technique The effectiveness of this approach is evaluated using coordinated scan dataset discussed in *Chapter 4* and benchmark scan dataset.

- In *Chapter 6*, a clustering and outlier-based approach for network traffic anomaly detection is presented. To support the anomaly detection process, we introduce a tree-based clustering algorithm to generate a set of reference points. Finally, it uses our outlier score function to test each candidate object to identify network anomalies using estimated score values. The proposed technique is validated using several real-life and benchmark datasets

- *Chapter 7* introduces an unsupervised approach for network anomaly detection to detect known as well as unknown attacks without any prior knowledge. We propose an unsupervised tree-based clustering technique with a stability measure and a clustering labelling technique based on multiple objectives to label each cluster followed by clustering. This method is validated using several real-life datasets to detect known as well as unknown attacks.

- *Chapter 8* initially evaluates the performance of different information entropy metric. viz , Shannon entropy, generalized entropy, and Renyi's entropy to detect DDoS flooding attacks We propose an extended entropy metric-based approach for detecting all possible scenarios of DDoS flooding attacks at victim-end and extended to selective ensemble to improve the detection rate. This approach is validated using several real-life datasets.

- Finally, this thesis is concluded in *Chapter 9* summarizing the work described in detail in the previous chapters The chapter also presents future research directions in the network anomaly detection domain.

Chapter 2

Background

This chapter includes three parts, viz., basics of network anomalies, detection of such anomalies and evaluation criteria for detection methods. It presents definition, causes, sources, types of anomalies in network or host and detection approaches. We also discuss common evaluation criteria to measure performance of network anomaly detection methods or systems and concludes with a summary.

2.1 Anomalies in Network

Anomalies are instances in data that do not conform the normal behavior. The instances are also known as objects, points, events, vectors, or samples. Anomalies in network can be defined as any network events or operations that deviate from normal network behavior. They may occurs due to several reasons including (i) malicious activities that interpret normal network services, (ii) network overload, (iii) device malfunctioning, and (iv) compromises in different network parameters. Network anomalies are broadly categorized into two types [18] (a) performance related anomalies and (b) security related anomalies. We discuss each of them with sources and causes in details.

2.1.1 Performance Related Anomalies

These anomalies normally result in network or system failures or performance degradation. Typical examples of performance related anomalies are: file server failures, paging across the network, broadcast storms, babbling node, and transient con-

gestion Performance related anomalies may occur due to *vulnerabilities* involved in a network or a system In a network based system, vulnerabilities are inherent weaknesses in the design, implementation and management of the system, that render the systems susceptible to a threat [36]. It is not possible to list definite sources of network based system vulnerabilities Most common sources of network vulnerabilities are pointed out below.

(a) *Poor design*: Lack of appropriate design in a hardware and software system may lead to threat to the system For example, the sendmail flaw in earlier versions of UNIX enabled hackers to gain privileged access to the system.

(b) *Incorrect implementation*. Incorrect or erroneous configuration of the system may also lead to vulnerabilities due to lack of inexperience, insufficient training, or sloppy work. For example, configuring a system that does not have restricted-access privileges on system files, may allow these files to be altered by unauthorized users.

(c) *Poor security management*: Use of inadequate management of procedures are another sources of network vulnerabilities. For example, lack of guarantee that security procedures are being followed and that no single person has total control of a system.

(d) *Internet technology vulnerability*: Internet technology has been and continues to be vulnerable There are reports of all sorts of loopholes, weaknesses, and gaping holes in both software and hardware technologies every day by. Such vulnerabilities have led to attacks such as CodeRed worm, Slammer worm, etc.

(e) *The nature of intruder activity*: Hacker technologies are developing faster than the rest of the technology and are flourishing. For example, $W32/Mydoom.n@M$ $M!1812A23B5D92$ is a new malware threat[1].

(f) *The difficulty of fixing vulnerable systems*: It is often difficult to fix a vulnerable systems within stipulated time period. There is concern about the ability of system administrators to cope with the number of patches issued for system

[1]http://www.mcafee.com/us/mcafee-labs.aspx

vulnerabilities. As the number of vulnerabilities rises, system and network administrators face a more difficult situation.

(g) *Social engineering*: Social engineering involves a hacker's use of psychological tricks on legitimate users of a computer system, in order to gain information such as username and password that one needs to enter into the system. So, social engineering can be easily used to transform a critical security hole to a potential threat.

Thus, there are many causes for traffic anomalies which arise when attackers exploit network-based vulnerabilities. They are difficult to fix in real-time. Some of vulnerabilities are: network configuration, hardware, parameter, logging and monitoring, communication and wireless vulnerabilities. In this subsections we have discussed performance related network anomalies, possible sources and causes of network vulnerabilities. Security related anomalies is our prime focus in this thesis, discussed below.

2.1.2 Security Related Anomalies

Security related anomalies occur while the network traffic does not follow normal behavior. Network attack categories include denial of service (DoS), distributed DoS, probe, user to root, remote to local and coordinated scan attacks. The main causes of network attacks or intrusions are malicious entities, who hijack network bandwidth by flooding the network with unnecessary traffic, thus starving other legitimate users. Malicious activities in a network is of various types such as point anomaly, contextual anomaly and collective anomaly [37], we describe each of them below.

(a) *Point Anomalies*: An instance of an individual data which has been found to be anomalous with respect to the rest of data then it is known as point anomaly. For example, credit card fraud.

(b) *Contextual Anomalies*: A data instance which has been found anomalous in a specific context is known as contextual anomaly. Context is induced by the

structure in the dataset. Two sets of attributes are used for defining a context (i) contextual, and (ii) behavioral attributes.

(c) *Collective Anomalies*: A collection of related data instances found to be anomalous with respect to the entire dataset are called Collective Anomalies. A collection of events is an anomaly but the individual events are not anomalies when they occur alone in the sequence.

We illustrate each type of anomaly in terms of fraudulent credit card transactions given in Figure 2.1.

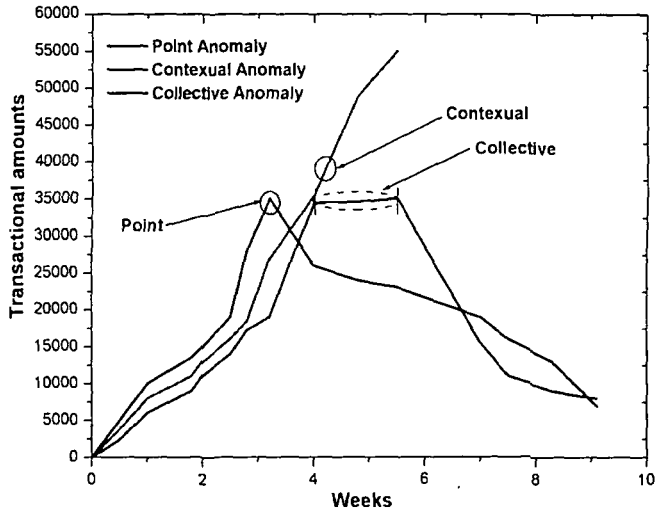


Figure 2.1: Illustration of point, contextual and collective anomaly

2.2 Attacks on Networks

A network attack exploits the vulnerability of a computer or network, and attempts to break into or compromise the security of the system. One who performs or attempts an attack or intrusion into a system is an attacker or intruder. Anderson [35] 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 to access the system as root or superuser. *Internal intruders* are further divided into *masquerade intruders* and *clandestine intruders*. 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. An 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.

2.3 Precursors to an Attack

The precursors to an attack are 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 the attacker's skill level, the type of vulnerability to exploit, prior knowledge, and starting location of the attacker. An attacker generally follows the steps shown in Figure 2.2 while launching an attack

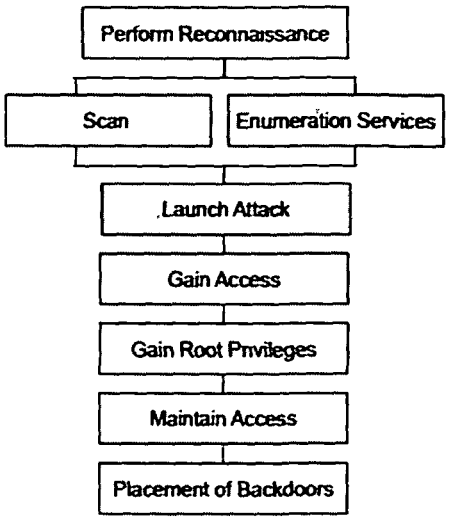


Figure 2.2: Steps in performing an attack

Performing reconnaissance means that the attacker uses certain techniques to gather information about the strength and positioning of enemy forces using *scanning* and enumeration of *services*. Once the vulnerabilities are identified, the attacker attempts to exploit them when *launching the attack*. The attacker can gain access as a user and then as root user to a system. Finally, to *place backdoors* on the system for further exploits, the attacker *maintains* access and cleans up any evidence left in the system.

2.4 Taxonomy of Network Attacks

An attack or intrusion is a sequence of operations to place a backdoors for exploitation in a network or a computer system Several network attack taxonomies are available in literature [38–41] The taxonomy of *intrusions* or *attacks* [42, 43] in computer systems that we adopt is summarized in Table 2 1

Table 2.1: Taxonomy of computer attacks characteristics and examples

Attack name	*Characteristics*	*Example*
Virus	(i) A self replicating program that infects the system without any knowledge or permission from the user (ii) Increases the infection rate of a network file system if the system is accessed by another computer	Trivial 88 D, Polyboot B, Tuareg
Worm	(i) A self replicating program that propagates through network services on computer systems without user intervention (ii) Can highly harm network by consuming network bandwidth	SQL Slammer, Mydoom, CodeRed Nimda
Trojan	(i) A malicious program that cannot replicate itself but can cause serious security problems in the computer system (ii) Appears as a useful program but in reality it has a secret code that can create a backdoor to the system, allowing it to do anything on the system easily, and can be called as the hacker gets control on the system without user permission	Mail Bomb, phishing attack
Denial of service (DoS)	(i) Attempts to block access to system or network resources (ii) The loss of service is the inability of a particular network or a host service, such as e-mail to function (iii) It is implemented by either forcing the targeted computer(s) to reset, or consuming resources (iv) Intended users can no longer communicate adequately due to non-availability of service or because of obstructed communication media	Buffer overflow, ping of death (PoD), TCP SYN, smurf, teardrop
Network Attack	(i) Any process used to maliciously attempt to compromise the security of the network ranging from the data link layer to the application layer by various means such as manipulation of network protocols (ii) Illegally using user accounts and privileges, performing actions to delete network resources and bandwidth, performing actions that prevent legitimate authorized users from accessing network services and resources	Packet injection, SYN flood
Physical Attack	An attempt to damage the physical components of networks or computers	Cold boot, evil maid
Password Attack	Aims to gain a password within a short period of time, and is usually indicated by a series of login failures	Dictionary attack, SQL injection attack
Information Gathering Attack	Gathers information or finds known vulnerabilities by scanning or probing computers or networks	SYS scan, FIN scan, XMAS scan
User to Root (U2R) attack	(i) It is able to exploit vulnerabilities to gain privileges of superuser of the system while starting as a normal user on the system (ii) Vulnerabilities include sniffing passwords, dictionary attack, or social engineering	Rootkit, loadmodule perl
Remote to Local (R2L) attack	(i) Ability to send packets to a remote system over a network without having any account on that system, gain access either as a user or as a root to the system and do harmful operations (ii) Performs attack against public services (such as HTTP and FTP) or during the connection of protected services (such as POP and IMAP)	Warezclient, warezmaster, imap ftp_write, multihop, phf, spy
Probe	(i) Scans the networks to identify valid IP addresses and to collect information about host (e g what services they offer, operating system used) (ii) Provides information to an attacker with the list of potential vulnerabilities that can later be used to launch an attack against selected systems and services	IPsweep, portsweep

Network attacks are also classified as *active* and *passive* Active attacks employ more overt actions on the network or system They can be much more devastating to a network. But passive attacks are designed to monitor and record traffic on the

network They are usually employed for gathering information that can be used later in active attacks. They are very difficult to detect, because there is no overt activity that can be monitored or detected. Examples of passive attacks are be packet sniffing or traffic analysis.

2.5 Traffic Monitoring and Analysis

Network traffic monitoring is usually performed at the intra-domain level in every large-scale autonomous system (AS) because the network topology is completely known and the AS is under the control of a single network operator, who can therefore manipulate his network and traffic without restrictions [44]. So, we deploy the monitoring systems in the intra-domain Internet to capture, analyze and decide whether an instance is normal or anomalous A view of a monitoring system deployment with DMZ[1] is shown in Figure 2.3.

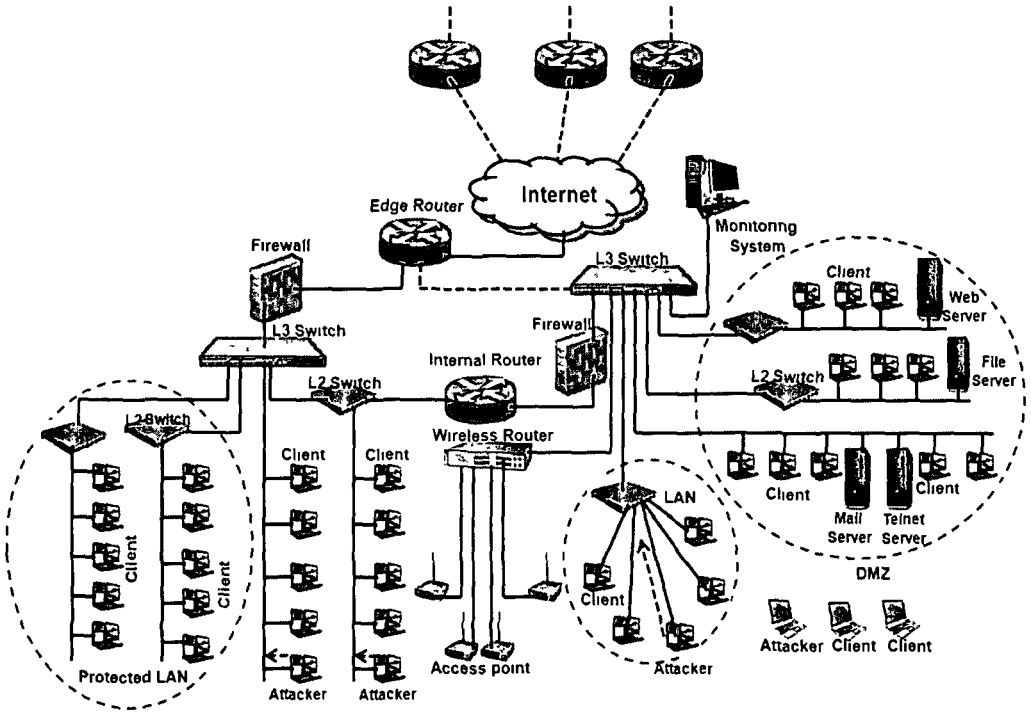


Figure 2.3: A typical view of monitoring system deployment with DMZ

[1]A demilitarized zone is a network segment located between a secured local network and unsecured external networks (Internet). A DMZ usually contains servers that provide services to users on the external network, such as Web, mail, and DNS servers. These servers must be hardened systems. Two firewalls are typically installed to form the DMZ.

2.6 Detection of Anomalies

Network anomaly detection is crucial for securing a network or a host. It allows for timely mitigation of anomalous traffic instances. Anomalies may be caused due to many reasons as discussed above Specifically, security related anomalies occur due to malicious activity (e g , scanning, denial of service, and probe) initiated by the attackers or intruders at different times. It leads to failure or misconfiguration of a network or a host A network anomaly detection system is introduced to capture, analysis and report alarms when an anomaly is detected and update the profiles of normal as well as attack instances.

2.6.1 Anomaly-based Network Intrusion Detection System (ANIDS)

An ANIDS is a system for detecting network anomalies by monitoring network traffic and classifying them as either normal or anomalous The classification is based on heuristics, rather than patterns or signatures, and attempts to detect any type of anomalies that falls out of normal system operation. It is able to detect known as well as unknown attacks without any prior knowledge.

2.6.2 Classification of ANIDSs

Intrusion detection has been studied for almost 20 years Intrusions can be detected because an intruder's behavior is noticeably different from that of a legitimate user. In addition, many unauthorized actions are detectable [45] ANIDSs are deployed as a second line of defense along with other preventive security mechanisms, such as user authentication and access control ANIDSs is classified into two types based on their deployment in real time.

A *host based IDS* (HIDS) monitors and analyzes the internals of a computing system rather than its external interfaces. It monitors all or parts of the dynamic behavior and the state of a computer system [46] A HIDS might detect internal activity such as which program accesses what resources and attempts illegitimate access. An example is a word processor that has suddenly and inexplicably starts

modifying the system password database. Similarly, a HIDS might look at the state of a system and stored information whether it is in RAM or in the file system or in log files or elsewhere. One can think of a HIDS as an agent that monitors whether anything or anyone internal or external has circumvented the security policy that the operating system tries to enforce.

A *network based IDS* (NIDS) deals with detecting intrusions in network traffic. Intrusions typically occur as anomalous patterns. Some techniques model the network traffic in a sequential fashion and detect anomalous sub-sequences [46]. The primary reason for these anomalies is attacks launched by outside attackers who want to gain unauthorized access to the network to steal information or to disrupt the network.

In a typical setting, a network is connected to the rest of the world through the Internet The NIDS reads all incoming packets or flows, trying to find suspicious patterns For example, if a large number of TCP connection requests to a very large number of different ports are observed within a short time, one could assume that there is someone committing a 'port scan' at some of the computer(s) in the network. Various kinds of port scans, and their launching tools are discussed in detail in [5]. Port scans mostly try to detect incoming shell codes in the same manner that an ordinary intrusion detection system does. In addition to inspecting the incoming network traffic, a NIDS also provides valuable information about intrusion from outgoing as well as local traffic. Some attacks might even be staged from inside of a monitored network or network segment, and therefore, not regarded as incoming traffic at all. The data available for intrusion detection systems can be at different levels of granularity, e.g , packet level traces, and IPFIX records. The data is high dimensional, typically, with a mix of categorical and continuous attributes.

Misuse-based intrusion detection normally searches for known intrusive patterns but anomaly based intrusion detection tries to identify unusual patterns. Today, researchers mostly concentrate on anomaly based network intrusion detection because it can detect known as well as unknown attacks.

There are several reasons that make intrusion detection a necessary part of the entire defense system. First, many traditional systems and applications were developed without security in mind. Such systems and applications were designed to

work in an environment, where security was never a major issue However, the same systems and applications when deployed in the current network scenario become major security headaches For example, a system may be perfectly secure when it is isolated but becomes vulnerable when it is connected to the Internet Intrusion detection provides a way to identify and thus allow response to attacks against these systems Second due to limitations of information security and software engineering practices, computer systems and applications may have design flaws or bugs that could be used by an intruder to attack systems or applications As a result certain preventive mechanisms (e g firewalls) may not be as effective as expected Intrusion detection techniques are classified into three types based on the detection mechanism [37, 47, 48] This classification scheme is described below

(a) *Misuse-based* This detection is based on a set of rules or signatures for known attacks and can detect all known attacks based on reference data How to write a signature that encompasses all possible variations of the pertinent attack is a challenging task

(b) *Anomaly-based* The principal assumption is that all intrusive activities are necessarily anomalous Such a method builds a *normal activity profile* and checks whether the system state varies from the established profile by a statistically significant amount to report intrusion attempts Anomalous activities that are not intrusive may be flagged as intrusive These are false positives One should select threshold levels so that the above two problems are unreasonably magnified Anomaly-based intrusion detection is computationally expensive because of overhead and the need to update several system profile matrices

(c) *Hybrid* This detection mechanism reaps benefits of both misuse and anomaly based detection techniques It also attempts to detect known as well as unknown attacks

In addition to the above an anomaly detection system works in any of four modes, viz , (i) supervised (ii) semi-supervised (iii) unsupervised and (iv) hybrid based on the availability of labeled data

Supervised ANIDS

A supervised ANIDS detects network anomalies using prior knowledge. It builds a predictive model for both normal and anomalous classes and compares any new instance with the predictive model to determine which class it belongs to Thus, to provide an appropriate solution in network anomaly detection, we need the concept of normal behavior of the network traffic. An event or an object is detected as anomalous if its degree of deviation with respect to the profile or behavior of the system, specified by the normality model, is high enough. We define a supervised system as follows.

Definition 2.6.1. *Let us consider an anomaly detection system I that uses a supervised approach It can be thought of as a pair $I = (M, D)$, where M is the model of normal behavior of the system and D is a proximity measure that allows one to compute, given an activity record, the degree of deviation that such activities have with regard to the model M Thus, each system has two main modules: (i) a modeling module and (ii) a detection module One trains the system for both normal and attack classes to obtain the model M. The obtained model is subsequently used by the detection module to evaluate new events or objects or traffic as normal or anomalous or outliers. In particular, the modeling module needs to be adaptive to cope with the dynamic scenarios.*

A generic architecture for a supervised ANIDS is given in Figure 2.4. A brief description of each component of the above system is given below.

(a) ***Traffic Capturing***. Traffic capturing is an important module in any NIDS. In this module, live network traffic is captured using the Libpcap [49] library, an open source C library offering an interface for capturing link-layer frames over a wide range of system architectures. It provides a high-level common Application Programming Interface to the different packet capture frameworks of various operating systems. The resulting abstraction layer allows programmers to rapidly develop highly portable applications

Libpcap defines a common standard format for files in which captured frames are stored, also known as the tcpdump format, currently a de facto standard widely used in public network traffic archives. Modern kernel-level capture frameworks on UNIX operating systems are mostly based on BSD or Berkeley Packet Filter (BPF)

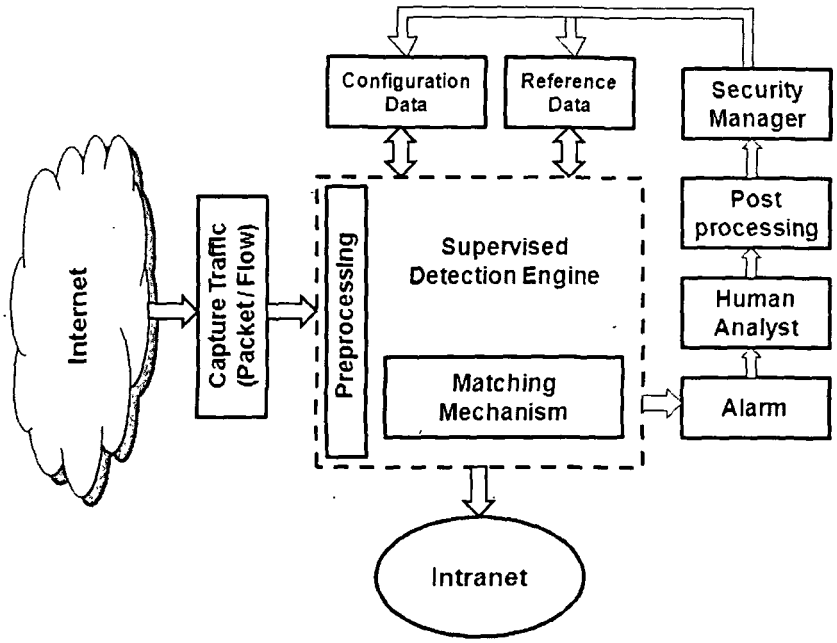


Figure 2.4: A generic architecture of supervised ANIDS

[50, 51]. BPF is a software device that taps network interfaces, copying packets into kernel buffers and filtering out unwanted packets directly in interrupt context. Definition of packets to be filtered can be written in a simple human readable format using Boolean operators and can be compiled into a pseudo-code and passed to the BPF device driver by a system call. The pseudo-code is interpreted by the BPF Pseudo-Machine, a lightweight high-performance state machine specifically designed for packet filtering. Libpcap also allows a programmer to write applications that transparently support a rich set of constructs to build detailed filtering expressions for most network protocols. A few Libpcap calls use these Boolean expressions, which can read directly from the user's command line, compile into pseudo-code and pass to Berkeley Packet Filter. Libpcap and BPF interact to allow network packet data to traverse several layers to finally be processed and transformed into in capture files (i.e., tcpdump format) or to samples for statistical analysis.

The raw network traffic is captured at both packet and flow levels. Packet level traffic can be captured using some popular tools, viz., Gulp (Lossless Gigabit Remote Packet Capture With Linux)[1] and Wireshark[2] and then preprocessed before

[1] http://staff.washington.edu/corey/gulp/
[2] http://www.wireshark.org/

sending to the detection engine. In addition, flow level traffic can be captured using some other tools, viz., NFDUMP[1], NFSEN[2], and ntop. The hierarchy of network traffic capturing components is given in Figure 2.5.

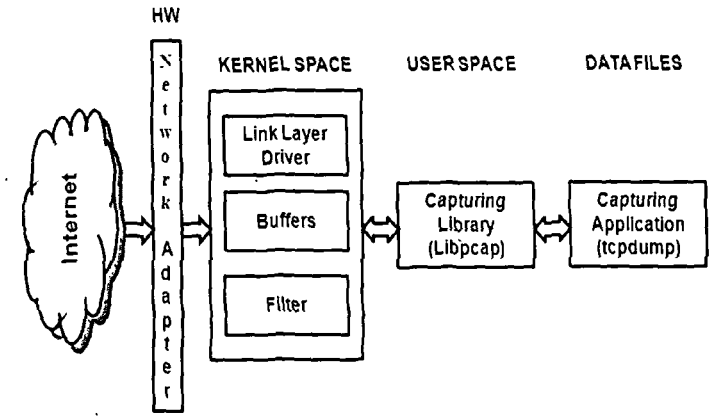


Figure 2.5: Hierarchy of network traffic capturing components

(b) ***Preprocessor***: In order to evaluate an IDS, an unbiased intrusion dataset in a standard format is required. Generally, a live captured packet contains a lot of raw data; some of them may not be relevant in the context of an IDS. Therefore, filtration of irrelevant parameters during capture and extraction of relevant features from the filtered data are important preprocessing functions of an IDS. In addition to these, data type conversion, normalization and discretization are also useful functions of this module depending on the anomaly detection mechanism used in the IDS.

(i) *Feature Extraction*: Feature extraction from raw data is an important step for anomaly based network intrusion detection. The evaluation of any intrusion detection algorithm on real time network data is difficult, mainly due to the high cost of obtaining proper labeling of network connections. The extracted features are of four types [42, 52] discussed as follows.

- *Basic features*: These can be derived from packet headers without inspecting the payload. The protocol type, service, flag, source bytes, and destination bytes are examples of some basic features.

[1]http://nfdump.sourceforge.net/
[2]http://nfsen.sourceforge.net/

- *Content based features* Domain knowledge is used to assess the payload of the original TCP (Transmission Control Protocol) packets. An example of this type of features is the number of failed login attempts.
- *Time-based features*: These features are estimated by capturing properties that hold over a T-second temporal time window One example of such a feature is the number of connections to the same host over the T-second time interval.
- *Connection-based features*: These features are computed over an historical window estimated over the last N packets An example of such feature is the number of packets flowing from source to destination.

(ii) *Data Type Conversion.* Both features and raw data may include numeric as well as categorical data For example, the protocol feature takes values such as tcp, icmp (Internet Control Message Protocol), telnet and udp. Therefore, to apply a clustering technique based on a proximity measure for either numeric or categorical data to detect network anomalies, it may be necessary to convert the data.

(iii) *Normalization*: In an intrusion dataset, all parameters or field values may not be equally weighted In such cases, normalization is considered useful before applying an anomaly detection mechanism

(iv) *Discretization*: The network intrusion data contains continuous valued attributes such as the number of packets the number of bytes, the duration of each connection, etc. These attributes may need to be transformed into binary features before applying any standard association mining algorithms. The transformation can be performed using a variety of supervised and unsupervised discretization techniques. For example, using the output scores of the anomaly detector as its ground truth, MINDS (Minnesota INtrusion Detection System) [53] employs a supervised binning strategy to discretize attributes. Initially, all distinct values of continuous attributes are put into one bin The worst bin in terms of purity is selected for partitioning until the desired number of bins is reached. The discretization of numeric attributes contributes to the comprehension of the final results.

These features are designed to assess attacks, which span intervals longer than 2 seconds. It is well known that features constructed from the data content of the connections, are more important when detecting R2L (Remote to Local) and U2R (User to Root) attack types in KDD99 intrusion dataset [52] The time based and connection based features are more important for detection of DoS (Denial of Service) and probing attack types [54].

(c) ***Anomaly Detection Engine***: This is the heart of any network anomaly detection system. It attempts to detect the occurrence of any intrusion either online or offline. In general, any network traffic data needs preprocessing before it is sent to the detection engine. If the attacks are known, they can be detected using a misuse detection approach. Unknown attacks can be detected with the anomaly based approach using an appropriate matching mechanism. The following are some important requirements that a matching mechanism must satisfy.

- Matching determines whether the new instance belongs to a known class defined by a high dimensional profile or not. Matching may be inexact. The membership of a test instance to a given pre-defined class represented by its profile, depends on (i) the proximity computed between the profile and the new test instance using a relevant subspace of features and (ii) a user-defined proximity threshold. Thus, the selection of an appropriate proximity measure and an appropriate threshold are crucial here.

- Matching must be fast

- Effective organization of the profiles may facilitate faster search during matching.

(d) ***Alarm***: This module is responsible for generation of alarm based on the indication received from the Anomaly Detection Engine. In addition to indicating the occurrence of an attack, alarms are useful for post diagnosis of the performance of the detection system. Alarms should indicate (i) the causes for the alarm to be raised, (ii) the source IP/Port address and target IP/Port address associated with the attack, and (iii) any background information to justify why it is a putative alarm.

(e) ***Human analyst***: A human analyst is responsible for analysis, interpretation and for taking necessary action based on the alarm information provided by the detection engine. The analyst also takes necessary steps to diagnose the alarm information as a post-processing activity to support reference or profile updation with the help of security manager.

(f) ***Post-processing***: This is an important module in a NIDS. This module processes the generated alarms for diagnosis of actual attacks. Appropriate post processing activities can help in reducing the false positive rate significantly.

(g) ***Security Manager***: Stored intrusion signatures are updated by the security manager (SM) as and when new intrusions become known. The analysis of novel intrusions is a highly complex task. The security manager has a multi-faceted role to play such as (i) to analyze alarm data, (ii) to recognize novel intrusion(s), and (iii) to update the signature or profile base.

(h) ***Reference Data***: The reference data stores information about signatures or profiles of known intrusions or normal behavior. Reference data must be stored in an efficient manner. Possible types of reference data used in the generic architecture of a NIDS are shown in Figure 2.6. In the case of ANIDS, it is mostly profiles. The processing elements update the profiles as new knowledge about the observed behavior becomes available. These updates are performed in a batch oriented fashion by resolving conflicts, if they arise.

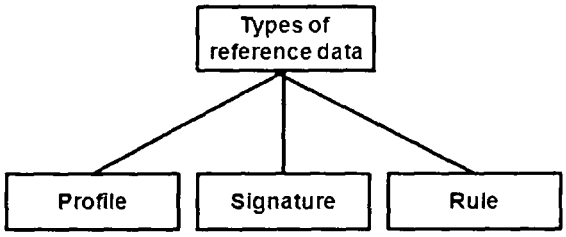


Figure 2.6: Types of reference data used in supervised ANIDS

Intermediate results such as partially created intrusion signatures are stored as *configuration data*. The space needed to store such information is usually quite large. The main steps for updation of configuration data are given in Figure 2.7. Intermediate results need to be integrated with existing knowledge to produce consistent, up-to-date results.

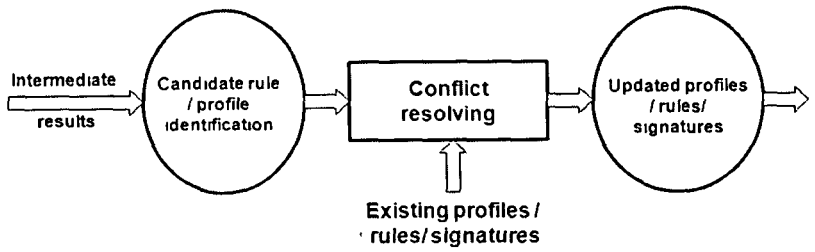


Figure 2.7: Steps for updation of configuration data in ANIDS

There are two major issues that arise in supervised anomaly detection. First, the anomalous instances are far fewer in number compared to normal instances in the training data. Issues that arise due to imbalanced class distributions have been addressed in the data mining literature [55]. Second, obtaining accurate and representative labels, especially for the anomaly class is usually challenging. A number of proposed techniques inject artificial anomalies in a normal dataset to obtain a labeled training dataset [56]. Other than these two issues, the supervised anomaly detection problem is similar to building predictive models.

Semi-supervised ANIDS

A semi-supervised ANIDS trains using labeled instances only for the normal class [57]. Since they do not require labels for the anomaly class, it is more readily used compared to supervised approaches We define a semi-supervised system as follows.

Definition 2.6.2. *Let I be a semi-supervised anomaly based detection system. It can be thought of as a pair $I = (M, D)$, where M is the model of normal behavior of the system and D is a proximity measure that allows one to compute, given an activity record, the degree of deviation that such activities have in regards to the model M. As discussed in the context of supervised ANIDS, each system has mainly two modules. The modeling module trains to get the normality model M and detect new traffic as normal or anomalous.*

For example, in spacecraft fault detection [58], an anomaly scenario may 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 behavior, and use the model to identify anomalies in the test data. However, semi-supervised learning uses normal data during training and the rest of the approach is the same

as supervised approach

Unsupervised ANIDS

An unsupervised ANIDS can be used for novel intrusion detection without prior knowledge and uses purely normal data Unsupervised network anomaly detection works well due to two major reasons (i) non-availability of labelled or purely normal data, and (ii) the expense of manual classification of a large volume of network traffic When collecting normal traffic data, it is extremely difficult to guarantee that there is no anomalous instance Clustering is a widely used method for unsupervised anomaly based intrusion detection [26,59–62] From classical data mining, we know that clustering is a method of grouping of objects based on similarity among the objects The similarity within a cluster is high whereas dissimilarity among clusters is high Clustering is a method of unsupervised study and analysis that is performed on unlabeled data [63] Unsupervised anomaly detection clusters test data into groups of similar instances which may be either normal or anomalous We define the unsupervised system as follows

Definition 2.6.3. *Let I be an unsupervised anomaly based detection system. It can be thought of as a pair $I = (M, D)$, where $M = \{G, A\}$, G represents groups of traffic based on proximity measure D, and A is the estimated score computed from each group The system I labels each traffic instance as normal or anomalous w r t the estimated score, A*

A generic architecture of an unsupervised ANIDS is given in Figure 2 8 This includes almost all the modules found in a supervised ANIDS except the anomaly detection engine and the labelling technique We discuss them below

(a) ***Unsupervised Engine*** This module is the heart of an anomaly detection system It consists of two modules viz , *detection* and *label* Based on the approach used, the *detection* module either groups similar instances or identifies exceptional instances in input data. The *label* module works after completion of the *detection* module to label each instance either as normal or anomalous based on the characteristics of each individual group such as size, compactness. the dominating subset of features and outlier score of each instance

(b) ***Labeling Strategy*** A clustering method merely groups the data without any

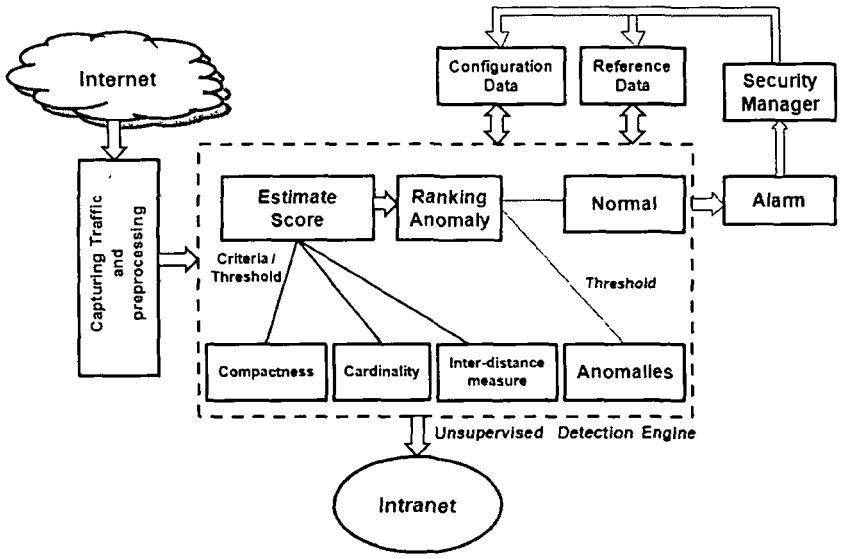


Figure 2.8: A generic architecture of unsupervised ANIDS

interpretation of the nature of the groups. To support appropriate interpretation of the groups labelling techniques are used. Labelling of clusters is a difficult issue. A labelling strategy typically makes the following assumptions [64].

- The number of normal instances vastly outnumbers the number of anomalous instances.
- Anomalies themselves are qualitatively different from normal instances.
- Intra-similarity among the instances of an anomalous group is higher than the same in a normal group of instances.

Unsupervised anomaly detection approaches work without any training data. In other words, these models are trained on unlabeled or unclassified data and they attempt to find intrusions lurking inside the data. The biggest 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 certain assumptions [64]. If these assumptions hold, intrusive instances can be identified based on characteristics of the group the instances belong to. However, these assumptions are not always true, especially in the case of DDoS attacks. Therefore, accurate labeling of an instance is a significant and crucial issue in an unsupervised ANIDS.

Hybrid ANIDS

A hybrid ANIDS combines both supervised and unsupervised approaches of network anomaly detection. Such approaches can detect known as well as unknown attacks. A hybrid approach attempts to identify known attacks based on a supervised model with reference to a set of training sample data using an appropriate matching mechanism. The test instances that neither belong to normal nor any of the known attack instances are handled by the unsupervised model for the identification of new normal or novel intrusions Several successful efforts have been made by researchers to develop hybrid ANIDSs [65–67]. A hybrid system is defined as follows.

Definition 2.6.4. *Let I be a hybrid anomaly based detection system. It can be thought of as a pair $I = (M, D)$, where $M = \{B, U\}$, B represents the supervised module that uses proximity measure D to detect known attacks, and U is the unsupervised module which uses estimated score computed from each group to detect unknown attacks.*

A generic architecture of a hybrid ANIDS is given in Figure 2.9. The modules in this architecture are the same as in supervised and unsupervised ANIDSs noted above except the detection engine. This detection engine is a combination of a supervised module and an unsupervised module As shown in the figure, the unsupervised module is used for only those undetected test instances forwarded by the supervised module. Once a novel intrusion is identified and confirmed, its reference (i.e., rule or signature) is built and inserted into the rule-base for the future reference of the supervised module.

The performance of an individual approach, either supervised or unsupervised, is not equally good for detection of all categories of attack as well as normal instances. There is the possibility of obtaining good detection accuracy for all categories in a dataset by using an appropriate combination of multiple well-performing detection approaches. The objective of such a combination is to provide the best performance from each participating approach for all attack classes. The selection of a supervised or unsupervised method at a particular level for a given dataset is a critical issue for the hybrid ANIDS.

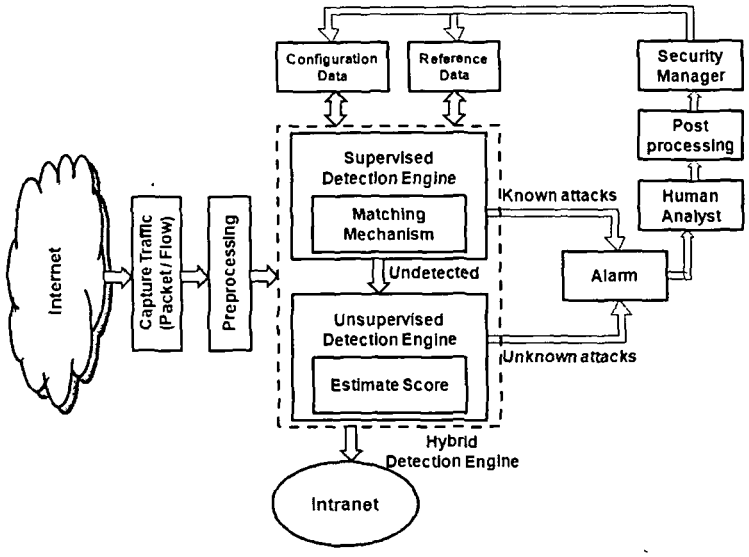


Figure 2.9: A generic architecture of a hybrid ANIDS

2.7 Aspects of Network Anomaly Detection

In this section, we present some important aspects of anomaly based network intrusion detection. The network intrusion detection problem is a classification or clustering problem formulated with the following components [37]: (i) types of input data, (ii) appropriateness of proximity measures, (iii) labelling of data, (iv) relevant feature identification and (v) reporting of anomalies. We discuss each of these topics in brief

2.7.1 Types of Input Data

A key aspect of any anomaly based network intrusion detection technique is the nature of the input data used for analysis. Input is generally a collection of data instances also referred to as objects, records, points, vectors, patterns, events, cases, samples, observations, entities [15]. Each data instance can be described using a set of attributes of binary, categorical or numeric type. Each data instance may consist of only one attribute (univariate) or multiple attributes (multivariate). In the case of multivariate data instances, all attributes may be of the same type or may be a mixture of data types. The nature of attributes determines the applicability of anomaly detection techniques.

2.7.2 Appropriateness of Proximity Measures

Proximity (similarity or dissimilarity) measures are necessary to solve many pattern recognition problems in classification and clustering. Distance is a quantitative degree of how far apart two objects are. Distance measures that satisfy metric properties [15] are simply called *metric* while other non-metric distance measures are occasionally called *divergence*. The choice of a proximity measure depends on the measurement type or representation of objects.

Generally, proximity measures are functions that take arguments as object pairs and return numerical values that become higher as the objects become more alike. A proximity measure is usually defined as follows

Definition 2.7.1. *A proximity measure D is a function $X \times X \rightarrow \mathbb{R}$ that has the following properties [68].*

- *Positivity:* $\forall_{x,y} \in X, D(x,y) \geq 0$
- *Symmetry:* $\forall_{x,y} \in X, D(x,y) = D(y,x)$
- *Maximality:* $\forall_{x,y} \in X, D(x,x) \geq D(x,y)$

where X is the data space (also called the universe*) and x, y are a pair of k-dimensional objects.*

The most common proximity measures for numeric [69–71], categorical [72] and mixed type [73] data are listed in Table 2.2. For numeric data, it is assumed that the data is represented as real vectors. The attributes take their values from a continuous domain. In Table 2.2, we assume that there are two objects, $x = x_1, x_2, x_3 \cdots x_d$, $y = y_1, y_2, y_3 \cdots y_d$ and $\sum^{-1}$ represents the data covariance with d number of attributes, i.e., dimensions.

For categorical data, computing similarity or proximity measures is not straightforward owing to the fact that there is no explicit notion of ordering among categorical values. The simplest way to find similarity between two categorical attributes is to assign a similarity of 1 if the values are identical and a similarity of 0 if the values are not identical. In the Table 2.2, $D_k(x_k, y_k)$ represents per-attribute similarity. The attribute weight w_k for attribute k is computed as shown in the table. Consider a categorical dataset X containing n objects, defined over a set

of d categorical attributes where A_k denotes the kth attribute $D_k(x_k, y_k)$ is the per-attribute proximity between two values for the categorical attribute A_k. Note that $x_k, y_k \in A_k$. In Table 2.2, IOF denotes *Inverse Occurrence Frequency* and OF denotes *Occurrence Frequency* [72]

Finally, mixed type data includes both categorical and numeric values. A common practice in clustering a mixed dataset is to transform categorical values into numeric values and then use a numeric clustering algorithm. Another approach is to compare the categorical values directly, in which two distinct values result in a distance of 1 while identical values result in a distance of 0. Of course, other measures for categorical data can be used as well Two well-known proximity measures, general similarity coefficient and general distance coefficient [73] for mixed type data are shown in Table 2.2. Such methods may not take into account the similarity information embedded in categorical values. Consequently, clustering may not faithfully reveal the similarity structure in the dataset [73, 74].

2.7.3 Labelling of Data

The label associated with a data instance denotes if that instance is normal or anomalous. It should be noted that obtaining accurate labeled data of both normal or anomalous types is often prohibitively expensive. Labeling is often done manually by human experts and hence substantial effort is required to obtain the labeled training dataset [37] Moreover, anomalous behavior is often dynamic in nature, e.g., new types of anomalies may arise, for which there is no labeled training data.

2.7.4 Relevant Feature Identification

Feature selection plays an important role in detecting network anomalies. Feature selection methods are 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. The feature selection process includes three major steps: (a) subset generation, (b) subset evaluation and (c) validation Three different approaches for subset generation are:

Table 2.2: Proximity measures for numeric, categorical and mixed type data

Numeric [69]			
Name	*Measure, $D_1(x_i, y_i)$*	*Name*	*Measure, $D_1(x_i, y_i)$*
Euclidean	$\sqrt{\sum_{i=1}^{d} \lvert x_i - y_i \rvert^2}$	Weighted Euclidean	$\sqrt{\sum_{i=1}^{d} a_i \lvert x_i - y_i \rvert^2}$
Squared Euclidean	$\sum_{i=1}^{d} \lvert x_i - y_i \rvert^2$	Squared-chord	$\sum_{i=1}^{d} (\sqrt{x_i} - \sqrt{y_i})^2$
Squared X^2	$\sum_{i=1}^{d} \frac{(x_i - y_i)^2}{x_i + y_i}$	City block	$\sum_{i=1}^{d} \lvert x_i - y_i \rvert$
Minkowski	$\sqrt[p]{\sum_{i=1}^{d} \lvert x_i - y_i \rvert^p}$	Chebyshev	$\max_i \lvert x_i - y_i \rvert$
Canberra	$\sum_{i=1}^{d} \frac{\lvert x_i - y_i \rvert}{x_i + y_i}$	Cosine	$\frac{\sum_{i=1}^{d} x_i y_i}{\sqrt{\sum_{i=1}^{d} x_i^2} \sqrt{\sum_{i=1}^{d} y_i^2}}$
Jaccard	$\frac{\sum_{i=1}^{d} x_i y_i}{\sum_{i=1}^{d} x_i^2 + \sum_{i=1}^{d} y_i^2 - \sum_{i=1}^{d} x_i y_i}$	Bhattacharyya	$-\ln \sum_{i=1}^{d} \sqrt{(x_i y_i)}$
Pearson	$\sum_{i=1}^{d} (x_i - y_i)^2$	Divergence	$2 \sum_{i=1}^{d} \frac{(x_i - y_i)^2}{(x_i + y_i)^2}$
Mahalanobis	$\sqrt{(x-y)^t \sum^{-1} (x-y)}$	-	-
Categorical [72]			
$w_k, k=1 \quad d$	*Measure, $D_k(x_k, y_k)$*	$w_k, k=1 \quad d$	*Measure $D_k(x_k, y_k)$*
$\frac{1}{2}$	$Overlap = \begin{cases} 1 & \text{if } x_k = y_k \\ 0 & \text{otherwise} \end{cases}$	$\frac{1}{d}$	$Eskin = \begin{cases} 1 & \text{if } x_k = y_k \\ \frac{n_k^2}{n_k^2+2} & \text{otherwise} \end{cases}$
$\frac{1}{d}$	$IOF = \begin{cases} 1 & \text{if } x_k = y_k \\ \frac{1}{1+\log f_k(x_k) \times \log f_k(y_k)} & \text{otherwise} \end{cases}$	$\frac{1}{d}$	$OF = \begin{cases} 1 & \text{if } x_k = y_k \\ \frac{1}{1+\log \frac{N}{f_k(x_k)} \times \log \frac{N}{f_k(y_k)}} & \text{otherwise} \end{cases}$
Mixed [73]			
Name	*Measure*	*Name*	*Measure*
General Similarity Coefficient	$D_{gsc}(x,y) = \frac{1}{\sum_{k=1}^{d} w(x_k, y_k)} \sum_{k=1}^{d} w(x_k, y_k) D(x_k, y_k)$ • For *numeric* attributes, $D(x_k, y_k) = 1 - \frac{\lvert x_k - y_k \rvert}{R_k}$ where R_k is the range of the k^{th} attribute $w(x_k, y_k) = 0$ if x or y has missing value for the k^{th} attribute, otherwise $w(x_k, y_k) = 1$ • For *categorical* attributes, $D(x_k, y_k) = 1$ if $x_k = y_k$, otherwise $D(x_k, y_k) = 0$, $w(x_k, y_k) = 0$ if data point x or y has missing value at k^{th} attribute otherwise $w(x_k, y_k) = 1$	General Distance Coefficient	$D_{gdc}(x,y) = \left(\frac{1}{\sum_{k=1}^{d} w(x_k, y_k)} \sum_{k=1}^{d} w(s_k, y_k) D^2(x_k, y_k) \right)^{\frac{1}{2}}$, where $D^2(x_k, y_k)$ is the squared distance for the k^{th} attribute, $w(x_k, y_k)$ is the same as in General Similarity Coefficient • For *numeric* attributes $D(x_k, y_k) = \frac{\lvert x_k - y_k \rvert}{R_k}$ where R_k is the range of k^{th} attribute • For *categorical* attributes $D(x_k, y_k) = 0$ if $x_k = y_k$ otherwise $D(x_k, y_k) = 1$

complete, *heuristic* and *random* Evaluation functions are categorized into five [75] distinct categories: score based, entropy or mutual information based, correlation based, consistency based and detection accuracy based. Simulation and real world implementation are the two ways to validate the evaluated subset. A conceptual framework of the feature selection process is shown in Figure 2.10.

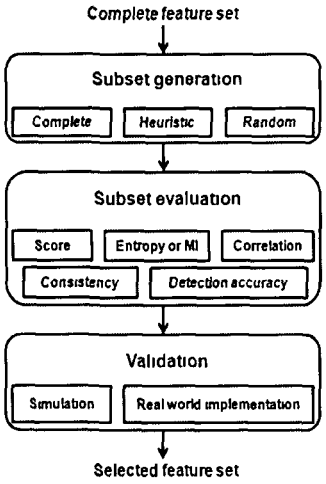


Figure 2.10: Framework of feature selection process

Feature selection algorithms have been classified into three types: *wrapper*, *filter* and *hybrid* methods [76]. 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. The hybrid feature selection method attempts to exploit the salient features of both wrapper and filter methods [76]

An example of wrapper-based feature selection method is [77], where the authors propose an algorithm to build a lightweight IDS by using modified Random Mutation Hill Climbing (RMHC) as a search strategy to specify a candidate subset for evaluation, and using a modified linear Support Vector Machine (SVM) based iterative procedure as a wrapper approach to obtain an optimum feature subset. The authors establish the effectiveness of their method in terms of efficiency in intrusion detection without compromising the detection rate. An example filter model for feature selection is [78], where the authors fuse correlation based and

minimal redundancy-maximal-relevance measures. They evaluate their method on benchmark intrusion datasets for classification accuracy Several other methods for feature selection are [42, 79–81]

2.7.5 Reporting Anomalies

An important aspect of any anomaly detection technique is the manner in which anomalies are reported [37]. Typically, the outputs produced by anomaly detection techniques are of two types: (a) a *score*, which is a value that combines (i) distance or deviation with reference to a set of profiles or signatures, (ii) influence of the majority in its neighborhood, and (iii) distinct dominance of the relevant subspace (as discussed in Section 2.7.4) (b) a *label*, which is a value (normal or anomalous) given to each test instance Usually the labelling of an instance depends on (i) the size of groups generated by an unsupervised technique, (ii) the compactness of the group(s), (iii) majority voting based on the outputs given by multiple indices (several example indices are given in Table 2.3), or (iv) distinct dominance of the subset of features.

2.8 Evaluation Criteria

We cannot have a method or a system which is totally or absolutely secure, without compromise. An evaluation of a method or a system in terms of accuracy or quality is basically a snapshot in time As time passes, new vulnerabilities may evolve, and current evaluation may become irrelevant However, information gathered during an evaluation process has an important role in shaping the final detection method or system. We discuss commonly used measures to evaluate network intrusion detection methods and systems Figure 2.11 shows the taxonomy of evaluation measures for network anomaly detection.

(a) ***Accuracy***: Accuracy is a metric that measures how correctly an IDS works, measuring the percentage of detection and failure as well as the number of false alarms that the system produces [94, 95]. If a system has 80% accuracy, it means that it correctly classifies 80 instances out of 100 to their actual classes. While there is a big diversity of attacks in intrusion detection, the main focus is that the

Table 2.3: Cluster validity measures

Reference	*Name of Index*	*Formula*	*Remark(s)*
Dunn [82]	Dunn Index	$DI = \frac{d_{min}}{d_{max}}$, where d_{min} denotes the smallest distance between two objects from different clusters d_{max} the largest distance within the same cluster	(i) Can identify dense and well-separated clusters (ii) High Dunn index is more desired for a clustering algorithm (iii) May not perform well with noisy data
Davies et al [83]	Davies Bouldin's index	$DB = \frac{1}{n}\sum_{i=1}^{n} {}_{i\neq j} max(\frac{\sigma_i+\sigma_j}{d(c_i,c_j)})$ where n is the number of clusters, σ_i is the average distance of all patterns in cluster i to their cluster center c_i, σ_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 between the cluster centers c_i and c_j	(i) Validation is performed using cluster quantities and features inherent to the dataset (ii) For compact clustering DB values should be as minimum as possible (iii) It is not designed to accommodate overlapping clusters
Hubert and Schultz [84]	C-index	$C = \frac{S-S_{min}}{S_{max}-S_{min}}$ where S is the sum of distances over all pairs of objects form the same cluster, n is the number of those pairs, S_{min} and S_{max} are the sum of n smallest distances and n largest distances, respectively	It needs to be minimized for better clustering
Baker and Hubert [85]	Gamma Index	$G = \frac{S+}{S+ + S-}$, where (S+) represents the number of times that a pair of samples not clustered together have a larger separation than a pair that were in the same clusters, $(S-)$ represents reverse outcome	This measure is widely used for hierarchical clustering
Rohlf [86]	G+ Index	$G+ = \frac{2(S-)}{n*(n-1)}$ where $(S-)$ is defined as for gamma index and n is the number of within cluster distances	It uses minimum value to determine the number of clusters in the data
Rousseeuw [87]	Silhouette Index	$SI = \frac{b_i-a_i}{max\{a_i,b_i\}}$, 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 cluster,	This index cannot be applied to datasets with sub-clusters
Goodman and Kruskal [88]	Goodman-Kruskal index	$GK = \frac{N_c-N_d}{N_c+N_d}$ where N_c and N_d are the numbers of concordant and disconcordant quadruples, respectively	(i) It is robust in outliers detection (ii) It requires high computation complexity in comparison to C-index
Jaccard [89]	Jaccard Index	$JI = \frac{a}{a+b+c}$ where a denotes the number of pairs of points with the same label in C and assigned to the same cluster in k, b denotes the number of pairs with the same label but in different clusters and c denotes the number of pairs in the same cluster, but with different class labels	It uses least information than Rand index measure
Rand [90]	Rand Index	$RI = \frac{a+d}{a+b+c+d}$, where d denotes the number of pairs with a different label in C that were assigned to a different cluster in k, rest are same with JI	It gives equal weights to false positives and false negatives during computation
Bezdek [91]	Partition coefficient	$PC = \frac{1}{n}\sum_{i=1}^{N}\sum_{j=1}^{n_c} u_{ij}^2$, where n_c is the number of clusters N is the number of objects in the dataset, u_{ij} is the degree of membership	(i) It finds the number of overlaps between clusters, (ii) It lacks connection with dataset
Bezdek [92]	Classification entropy	$CE = \frac{1}{N}\sum_{i=1}^{k}\sum_{j=1}^{n} u_{ij}log(u_{ij})$, same with partition coefficient	It measures the fuzziness of the cluster partitions
Xie and Beni [93]	Xie-Beni Index	$XB = \frac{\pi}{N\, d_{min}}$, where $\pi = \frac{\sigma_i}{n_i}$ is called compactness of cluster i Since n_i is the number of points in cluster i, σ_i is the average variation in cluster i $d_{min} = min\|k_i - k_j\|$	(i) It combines the properties of membership degree and the geometric structure of dataset (ii) Smaller XB means more compact and better separated clusters

system be able to detect an attack correctly. From real life experience, one can easily conclude that the actual percentage of abnormal data is much smaller than that of the normal [64, 96, 97]. Consequently, intrusions are harder to detect than normal traffic, resulting in excessive false alarms as the biggest problem facing IDSs. The following are some accuracy measures.

- *Sensitivity and Specificity*: These two measures [98] attempt to measure the accuracy of classification for a 2-class problem. When an IDS classifies data, its decision can be either right or wrong. It assumes true for right and false for wrong, respectively

  If S is a detector and D_t is the set of test instances, there are four possible

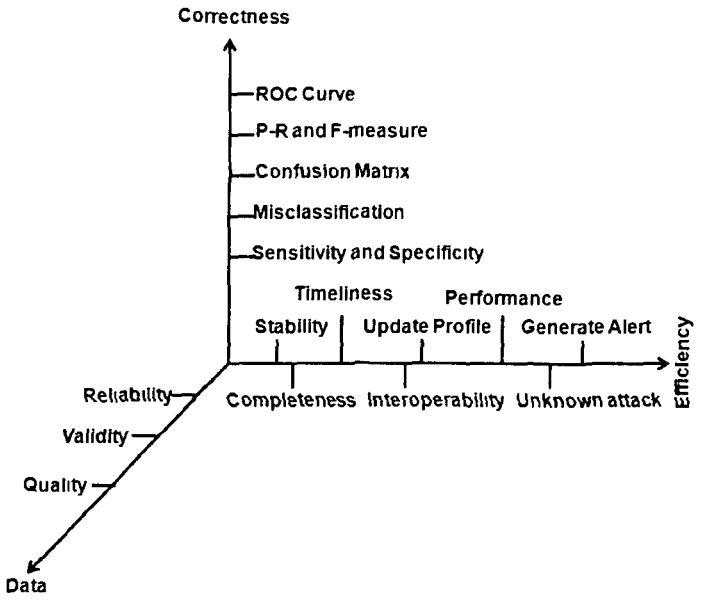


Figure 2.11: Taxonomy of evaluation measures

outcomes described using the confusion matrix given in Figure 2.12 When an anomalous test instance (p) is predicted as anomalous (Y) by the detector S, it is counted as true positive (TP); if it is predicted as normal (N), it is counted as false negative (FN). On the other hand, if a normal (n) test instance is predicted as normal (N) it is known as true negative (TN), while it is a false positive (FP) if it is predicted as anomalous (Y) [43, 98, 99]

The true positive rate (TPR) is the proportion of anomalous instances classified correctly over the total number of anomalous instances present in the test data. TPR is also known as *sensitivity* The false positive rate (FPR) is the proportion of normal instances incorrectly classified as anomalous over the total number of normal instances contained in the test data. The true negative rate (TNR) is also called *specificity*. TPR, FPR, TNR and the false negative rate (FNR) can be defined for the normal class. We illustrate all measures related to the confusion matrix in Figure 2.13.

Sensitivity is also known as the *hit rate* Between sensitivity and specificity, sensitivity is set at high priority when the system is to be protected at all cost, and specificity gets more priority when efficiency is of major concern [98]. Consequently, the aim of an IDS is to produce as many TPs and TNs

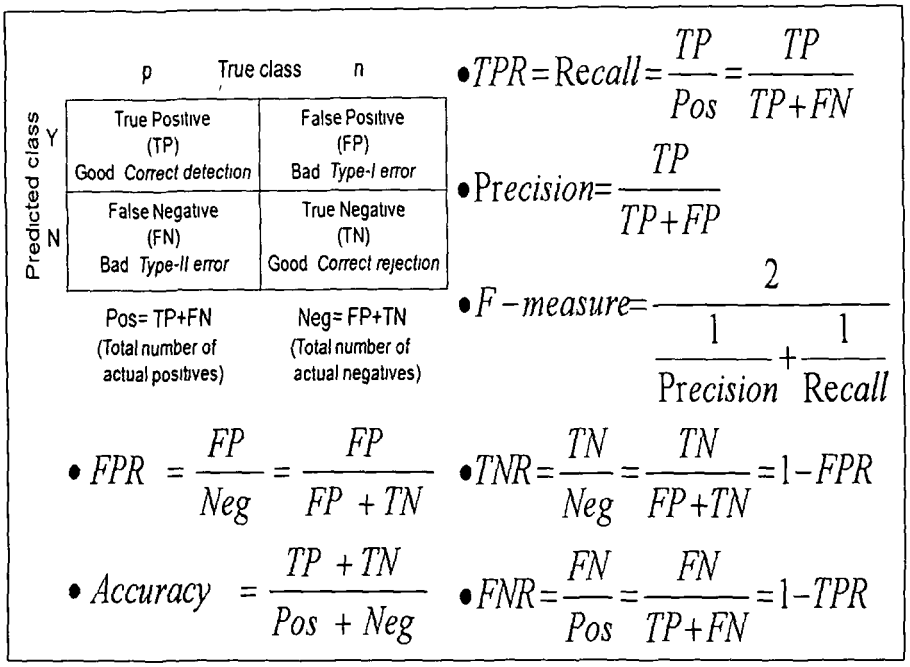


Figure 2.12: Confusion matrix and related evaluation measures

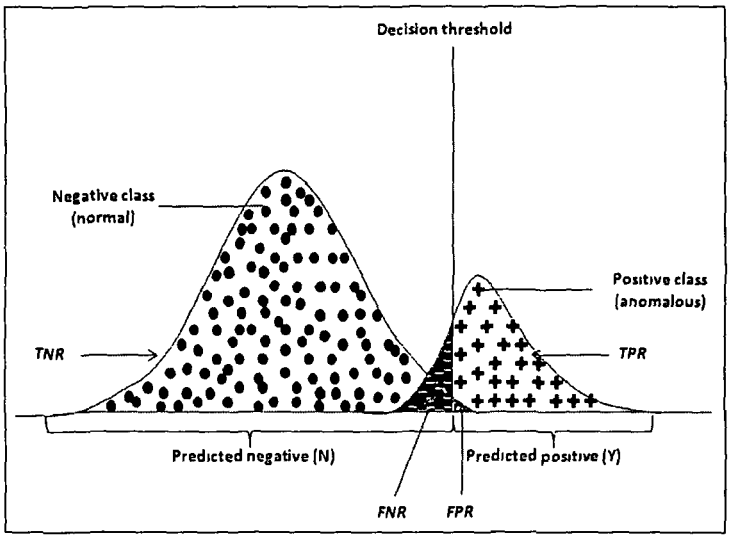


Figure 2.13: Illustration of confusion matrix in terms of their related evaluation measures

as possible while trying to reduce the numbers of both FPs and FNs. The majority of evaluation criteria use these variables and the relations among them to model the accuracy of the IDSs.

- *ROC Curves*: The Receiver Operating Characteristics (ROC) analysis orig-

inates from signal processing theory Its applicability is not limited only to intrusion detection, but extends to a large number of practical fields such as medical diagnosis, radiology, bioinformatics as well as artificial intelligence and data mining In intrusion detection, ROC curves are used on the one hand to visualize the relation between TP and FP rates of a classifier while tuning it and also to compare the accuracy with two or more classifiers The ROC space [100, 101] uses an orthogonal coordinate system to visualize classifier accuracy. Figure 2 14 illustrates the ROC approach normally used for network anomaly detection methods and systems evaluation.

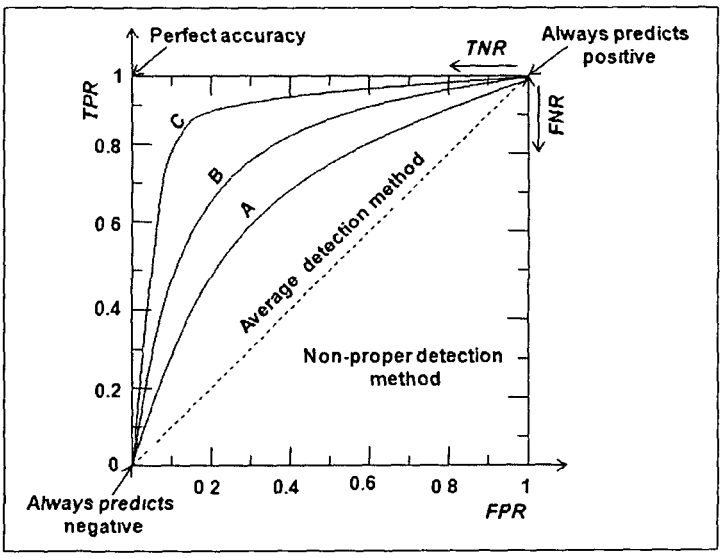


Figure 2.14: Illustration of ROC measure where A. B, C represents the accuracy in ascending order.

- *Misclassification rate.* This measure attempts to estimate the probability of disagreement between the true and predicted cases by dividing the sum of FN and FP by the total number of pairs observed, i.e., (TP+FP+FN+TN). In other words, misclassification rate is defined as (FN+FP)/(TP+FP+FN+TN).

- *Confusion Matrix*: The confusion matrix is a ranking method that can be applied to any kind of classification problem. The size of this matrix depends on the number of distinct classes to be detected The aim is to compare the actual class labels against the predicted ones as shown in Figure 2.12 The diagonal represents correct classification. The confusion matrix for intrusion

detection is defined as a 2-by-2 matrix, since there are only two classes known as *intrusion* and *normal* [43, 97, 99] Thus, the TNs and TPs that represent the correctly predicted cases lie on the matrix diagonal while the FNs and FPs are on the right and left sides As a side effect of creating the confusion matrix, all four values are displayed in a way that the relation between them can be easily understood

- *Precision, Recall and F-measure* Precision is a measure of how a system identifies attacks or normals A flagging is accurate if the identified instance indeed comes from a malicious user, which is referred to as true positive The final quantity of interest is recall, a measure of how many instances are identified correctly (see Figure 2 12) Precision and recall are often inversely proportional to each other and there is normally a trade-off between these two ratios An algorithm that produces low precision and low recall is most likely defective with conceptual errors in the underlying theory The types of attacks that are not identified can indicate which areas of the algorithm need more attention Exposing these flaws and establishing the causes assist future improvement

  The F-measure mixes the properties of the previous two measures as the harmonic mean of precision and recall [43, 99] If we want to use only one accuracy metric as an evaluation criterion, F-measure is the most preferable Note that when precision and recall both reach 100%, the F-measure is the maximum, i e , 1 meaning that the classifier has 0% false alarms and detects 100% of the attacks Thus a good classifier is expected to obtain F-measure as high as possible

(b) ***Performance*** The evaluation of an IDS s performance is an important task It involves many issues that go beyond the IDS itself Such issues include the hardware platform, the operating system or even the deployment of the IDS For a NIDS, the most important evaluation criterion for its performance is the system s ability to process traffic on a high speed network with minimum packet loss while working in real time In real network traffic, the packets can be of various sizes, and the effectiveness of a NIDS depends on its ability to handle packets of any size

In addition to the processing speed, the CPU and memory usage can also serve as measurements of NIDS performance [102]. These are usually used as indirect measures that take into account the time and space complexities of intrusion detection algorithms Finally, the performance of any NIDS is highly dependant upon (i) its individual configuration, (ii) the network it is monitoring, and (iii) its position in that network.

(c) ***Completeness***: The completeness criterion represents the space of the vulnerabilities and attacks that can be covered by an IDS. This criterion is very hard to assess because having omniscience of knowledge about attacks or abuses of privilege is impossible. The completeness of an IDS is judged against a complete set of known attacks. The ability of an IDS is considered complete, if it covers all the known vulnerabilities and attacks

(d) ***Timeliness***: An IDS that performs its analysis as quickly as possible enables the human analyst or the response engine to promptly react before much damage is done within a specific time period This prevents the attacker from subverting the audit source or the IDS itself The response generated by the system while combating an attack is very important. Since the data must be processed to discover intrusions, there is always a delay between the actual moment of the attack and the response of the system. This is called *total delay*. Thus, the total delay is the difference between t_{attack} and $t_{response}$. Thus smaller the total delay, the better an IDS is with respect to its response. No matter if an IDS is anomaly based or signature based, there is always a gap between the starting time of an attack and its detection.

2.9 Summary

In this chapter, we introduced basics of network anomalies and anomalies that commonly arise in networks. We explain two major categories of network anomalies, viz., performance related anomalies and security related anomalies. We described different network vulnerabilities that exist with their sources. An attacker exploits these vulnerabilities to cause network failure or degrade performance. In addition, we discuss sources of security related anomalies, types of network attacks, steps to

launch an attack and a taxonomy of attacks We also introduce the prime category of network anomaly detection methods with architecture, components, pros and cons.Finally, we present various measures that are normally used in evaluation of network anomaly detection methods and systems. These measures are used in evaluation of our detection methods discussed in subsequent chapters in the thesis.

Chapter 3

Related Work

This chapter starts with an overview of network anomaly detection, discusses existing methods and systems under six major categories We also include a list of tools with their features that are used by the network defenders and IDS developer during the execution of different steps to analyze network traffic. It concludes with a list of recommendations for the defenders as well as developers and a summary.

3.1 Introduction

Due to advancements in Internet technologies and the concomitant rise in the number of network attacks, network intrusion detection has become a significant research issue. In spite of remarkable progress and a large body of work, there are still many opportunities to advance the state-of-the-art in detecting and thwarting network based attacks [47].

The term *anomaly-based intrusion detection in networks* refers to the problem of finding exceptional network traffic patterns that do not conform to the expected normal behavior. These nonconforming patterns are often referred to as anomalies, outliers, exceptions, aberrations, surprises, peculiarities or discordant observations in various application domains [37, 103] Out of these, anomalies and outliers are two of the most commonly used terms in the context of anomaly based intrusion detection in networks.

The statistics community has been studying the problem of detection of anomalies or outliers from as early as the 19th century [104]. In recent decades, machine

learning has started to play a significant role in anomaly detection A good number of anomaly based intrusion detection techniques in networks have been developed by researchers Many techniques work in specific domains although others are more generic

3.1.1 Prior Surveys on Network Anomaly Detection

Network anomaly detection is a broad research area, which already boasts a number of surveys, review articles, as well as books An extensive survey of anomaly detection techniques developed in machine learning and statistics has been provided by [105, 106] Agyemang et al [107] present a broad review of anomaly detection techniques for numeric as well as symbolic data An extensive overview of neural networks and statistics based novelty detection techniques is found in [108] Patcha and Park [11] and Snyder [109] present surveys of anomaly detection techniques used specifically for cyber intrusion detection

A good amount of research on outlier detection in statistics is found in several books [110–112] as well as survey articles [113–115] Exhaustive surveys of anomaly detection in several domains have been presented in [37 116] Callado et al [117] report major techniques and problems identified in IP traffic analysis, with an emphasis on application detection Zhang et al [118] present a survey on anomaly detection methods in networks A review of flow based intrusion detection is presented by Sperotto et al [119] who explain the concepts of flow and classified attacks, and provide a detailed discussion of detection techniques for scans, worms, Botnets and DoS attacks

An extensive survey of DoS and DDoS attack detection techniques is presented in [120] Discussion of coordinated systems design and security for network is found in [121, 122] Wu and Banzhaf [13] present an overview of applications of computational intelligence methods to the problem of intrusion detection They include various methods such as artificial neural networks fuzzy systems, evolutionary computation, artificial immune systems, swarm intelligence, and soft computing A general comparison of various survey works available in the literature with our work is shown in Table 3 1

3.1.2 Motivation and Contributions

Even though there are several surveys available in the literature on network anomaly detection [11 37 116] surveys such as [11 116], discuss far fewer detection methods than we do in this chapter In [37], the authors discuss anomaly detection in general and cover the network intrusion detection domain only briefly None of the surveys [11, 37, 116] include common tools used during execution of various steps in network anomaly detection They also do not discuss approaches that combine several individual methods to achieve better performance In this chapter, we present a structured and comprehensive survey on anomaly based network intrusion detection in terms of general overview, methods, systems, and tools with a discussion of challenges and recommendations The major contributions of the survey presented in this chapter are the following

(a) Like the categorization of the network anomaly detection research suggested in [11, 37, 105 107], we classify detection methods and NIDSs into a number of categories In addition, we also provide an analysis of many methods in terms of their capability and performance datasets used, matching mechanism, number of parameter, and detection mechanism

(b) Most existing surveys do not cover ensemble approaches or data fusion for network anomaly detection but we do

3.2 Methods and Systems for Network Anomaly Detection

The classification of network anomaly detection methods and systems that we adopt is shown in Figure 3 1 This scheme is based on the nature of algorithms used It is not straightforward to come up with a classification scheme for network anomaly detection methods and systems, primarily because there is substantial overlap among the methods used in the various classes in any particular scheme we may adopt We have decided on six distinct classes of methods and systems We call them *statistical, classification based, clustering and outlier based, soft computing knowledge-based* and *combination learners* Most methods have subclasses as given in Figure

Table 3.1: A generic comparison of our survey with existing survey articles

Methods /NIDSs /Tools	*Topics covered*	[105]	[107]	[108]	[11]	[113]	[114]	[37]	[116]	[119]	[120]	[13]	[123]	[124]	[125]	[53]	*Our survey*
Methods	Statistical	√	√	√	√	√	√	√	√		√	√	√		√		√
	Classification-based	√	√		√		√	√	√	√		√	√	√			√
	Knowledge-based							√	√	√	√			√		√	√
	Soft computing												√		√		√
	Clustering-based	√	√	√	√			√	√				√				√
	Ensemble-based																√
	Fusion-based																√
	Hybrid																√
NIDSs	Statistical								√						√		√
	Classification-based																√
	Soft computing														√		√
	Knowledge-based								√							√	√
	Data Mining								√					√		√	√
	Ensemble-based																√
	Hybrid																√
Tools	Capturing, Preprocessing, Attack launching																√

(c) Most existing surveys avoid feature selection methods, which are crucial in the network anomaly detection task We present several techniques to determine feature relevance in intrusion datasets and compare them

(d) In addition to discussing detection methods, we provide several NIDSs with architectures for a few, their components and functionalities, and also present a comparison among existing NIDSs.

(e) We summarize tools used in various steps for network traffic anomaly detection.

3.1. Figure 3.2 shows the approximate statistics of papers published during the period 2000 to 2012 in each category.

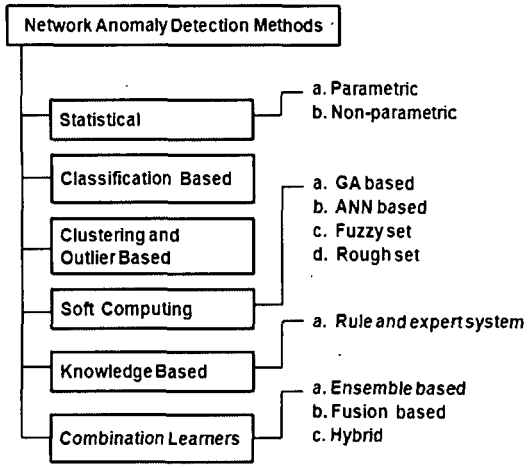


Figure 3.1: Classification of network anomaly detection methods (GA-Genetic Algorithm, ANN-Artificial Neural Network)

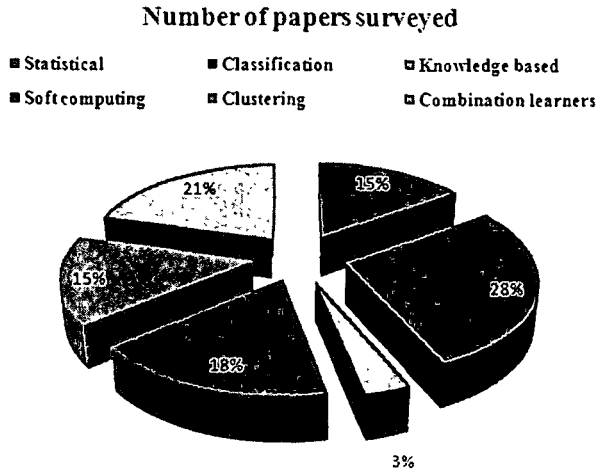


Figure 3.2: Statistics of the surveyed papers during the year 2000 to 2012

We distinguish between network anomaly detection methods and systems in this work, although such a distinction is difficult to make sometimes. A network intrusion detection system (NIDS) usually integrates a network intrusion detection method within an architecture that comprises other associated sub-systems to build a stand-alone practical system that can perform the entire gamut of activities needed for intrusion detection. We present several NIDSs with their architectures and components as we discuss various anomaly detection categories.

3.2.1 Statistical Methods and Systems

Statistically speaking, an anomaly is an observation which is suspected of being partially or wholly irrelevant because it is not generated by the stochastic model assumed [126]. Normally, statistical methods fit a statistical model (usually for normal behavior) to the given data and then apply a statistical inference test to determine if an unseen instance belongs to this model. Instances that have a low probability to be generated from the learnt model based on the applied test statistic are declared anomalies. Both parametric and nonparametric techniques have been applied to design statistical models for anomaly detection. While parametric techniques assume knowledge of the underlying distribution and estimate the parameters from the given data [127], nonparametric techniques do not generally assume knowledge of the underlying distribution [128].

An example of a statistical IDS is HIDE [125]. HIDE is an anomaly based network intrusion detection system, that uses statistical models and neural network classifiers to detect intrusions HIDE is a distributed system, which consists of several tiers with each tier containing several Intrusion Detection Agents (IDAs). IDAs are IDS components that monitor the activities of a host or a network. The probe layer (i.e., top layer as shown in Figure 3.3) collects network traffic at a host or in a network, abstracts the traffic into a set of statistical variables to reflect network status, and periodically generates reports to the event preprocessor The event preprocessor layer receives reports from both the probe and IDAs of lower tiers, and converts the information into the format required by the statistical model. The statistical processor maintains a reference model of typical network activities, compares reports from the event preprocessor with the reference models, and forms a stimulus vector to feed into the neural network classifier. The neural network classifier analyzes the stimulus vector from the statistical model to decide whether the network traffic is normal. The post-processor generates reports for the agents at higher tiers. A major of attraction of HIDE is its ability to detect UDP flooding attacks even with attack intensity as low as 10% of background traffic.

Of the many statistical methods and NIDSs [127, 129–137], only a few are described below in brief.

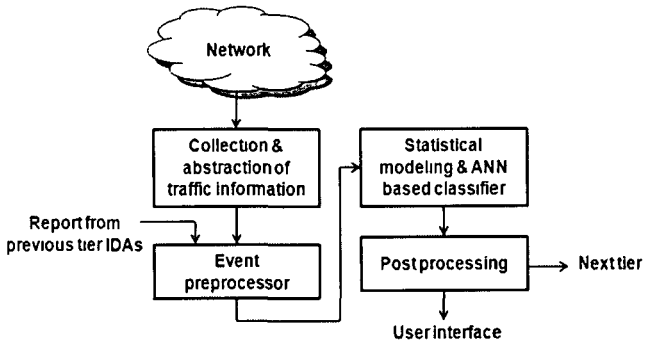


Figure 3.3: Architecture of HIDE system

Bayesian networks [138] are capable of detecting anomalies in a multi-class setting. Several variants of the basic technique have been proposed for network intrusion detection and for anomaly detection in text data [37]. The basic technique assumes independence among different attributes. Several variations of the basic technique that capture the conditional dependencies among different attributes using more complex Bayesian networks have also been proposed. For example, the authors of [139] introduce an event classification based intrusion detection scheme using Bayesian networks The Bayesian decision process improves detection decision to significantly reduce false alarms. Manikopoulos and Papavassiliou [129] introduce a hierarchical multi-tier multi-window statistical anomaly detection system to operate automatically, adaptively, and pro-actively It applies to both wired and wireless ad-hoc networks This system uses statistical modeling and neural network classification to detect network anomalies and faults. The system achieves high detection rate along with low misclassification rate when the anomaly traffic intensity is at 5% of the background traffic but the detection rate is lower at lower attack intensity levels such as 1% and 2%.

Association rule mining [140], conceptually a simple method based on counting of co-occurrences of items in transactions databases, has been used for one-class anomaly detection by generating rules from the data in an unsupervised fashion. The most difficult and dominating part of an association rule discovery algorithm is to find the itemsets that have strong support Mahoney and Chan [131] present an algorithm known as LERAD that learns rules for finding rare events in time-series data with long range dependencies and finds anomalies in network packets over

TCP sessions. LERAD uses an Apriori-like algorithm [140] that finds conditional rules over nominal attributes in a time series, e.g., a sequence of inbound client packets. The antecedent of a created rule is a conjunction of equalities, and the consequent is a set of allowed values, e.g., *if port=80 and word3=HTTP/1.0 then word1=GET or POST*. A value is allowed if it is observed in at least one training instance satisfying the antecedent. The idea is to identify rare anomalous events: those which have not occurred for a long time and which have high anomaly score. LERAD is a two-pass algorithm. In the first pass, a candidate rule set is generated from a random sample of training data comprised of attack-free network traffic. In the second pass, rules are trained by obtaining the set of allowed values for each antecedent.

A payload-based anomaly detector for intrusion detection known as PAYL is proposed in [132]. PAYL attempts to detect the first occurrence of a worm either at a network system gateway or with an internal network from a rogue device and to prevent its propagation. It employs a language-independent n-gram based statistical model of sampled data streams. In fact, PAYL uses only a 1-gram model (i.e., it looks at the distribution of values contained within a single byte) which requires a linear scan of the data stream and a small 256-element histogram. In other words, for each ASCII character in the range 0-255, it computes its mean frequency as well as the variance and standard deviation. Since payloads (i.e., arriving or departing contents) at different ports differ in length, PAYL computes these statistics for each specific observed payload length for each port open in the system. It first observes many exemplar payloads during the training phase and computes the payload profiles for each port for each payload length. During detection, each incoming payload is scanned and statistics are computed. The new payload distribution is compared against the model created during training. If there is a significant difference, PAYL concludes that the packet is anomalous and generates an alert. The authors found that this simple approach works surprisingly well.

Song et al. [133] propose a conditional anomaly detection method for computing differences among attributes and present three different expectation-maximization algorithms for learning the model. They assume that the data attributes are parti-

tioned into *indicator* attributes and *environmental* attributes based on the decision taken by the user regarding which attributes indicate an anomaly The method learns the typical indicator attribute values and observes subsequent data points, and labels them as anomalous or not based on the degree the indicator attribute values differ from the usual indicator attribute values However, if the indicator attribute values are not conditioned on environmental attributes values, the indicator attributes are ignored effectively The precision/recall of this method is greater than 90 percent

Lu and Ghorbani [135] present a network signal modeling technique for anomaly detection by combining wavelet approximation and system identification theory They define and generate fifteen relevant traffic features as input signals to the system and model daily traffic based on these features The output of the system is the deviation of the current input signal from the normal or regular signal behavior Residuals are passed to the IDS engine to take decisions and obtain 95% accuracy in the daily traffic In addition, a nonparametric adaptive cumulative sum (CUSUM) method for detecting network intrusions is discussed at [137]

In addition to the detection methods, there are several statistical NIDSs As mentioned earlier, a NIDS includes one or more intrusion detection methods that are integrated with other required sub-systems necessary to create a practical suitable system We discuss a few below

N@G (Network at Guard) [141] is a hybrid IDS that exploits both misuse and anomaly approaches N@G has both network and host sensors Anomaly based intrusion detection is pursued using the chi-square technique on various network protocol parameters It has four detection methodologies, viz data collection, signature based detection, network access policy violation and protocol anomaly detection as a part of its network sensor It includes audit trails, log analysis, statistical analysis and host access policies as components of the host sensor The system has a separate IDS server, i e a management console to aggregate alerts from the various sensors with a user interface, a middle-tier and a data management component It provides real time protection against malicious changes to network settings on client computers which includes unsolicited changes to the Windows Hosts file and Windows Messenger service

Flow-based statistical aggregation scheme (FSAS) [142] is a flow-based statistical IDS. It comprises of two modules: *feature generator* and *flow-based detector*. In the feature generator, the event preprocessor module collects the network traffic of a host or a network. The event handlers generate reports to the flow management module. The flow management module efficiently determines if a packet is part of an existing flow or it should generate a new flow key. By inspecting flow keys, this module aggregates flows together, and dynamically updates per-flow accounting measurements. The event time module periodically calls the feature extraction module to convert the statistics regarding flows into the format required by the statistical model. The neural network classifier classifies the score vectors to prioritize flows with the amount of maliciousness. The higher the maliciousness of a flow, the higher is the possibility of the flow being an attacker.

Advantages of statistical network anomaly detection include the following: (i) They do not require prior knowledge of normal activities of the target system. Instead, they have the ability to learn the expected behavior of the system from observations. (ii) Statistical methods can provide accurate notification or alarm generation of malicious activities occurring over long periods of time, subject to setting of appropriate thresholding or parameter tuning. (iii) They analyze the traffic based on the theory of abrupt changes, i.e., they monitor the traffic for a long time and report an alarm if any abrupt change (i.e., significant deviation) occurs.

Drawbacks of the statistical model for network anomaly detection include the following. (i) They are susceptible to being trained by an attacker in such a way that the network traffic generated during the attack is considered normal. (ii) Setting the values of the different parameters or metrics is a difficult task, especially because the balance between false positives and false negatives is an issue. Moreover, a statistical distribution per variable is assumed, but not all behaviors can be modeled using stochastic methods. Furthermore, most schemes rely on the assumption of a quasi-stationary process [11], which is not always realistic. (iii) It takes a long time to report an anomaly for the first time because the building of the models requires extended time. (iv) Several hypothesis testing statistics can be applied to detect anomalies. Choosing the best statistic is often not straightforward. In

particular, as stated in [136] constructing hypothesis tests for complex distributions that are required to fit high dimensional datasets is nontrivial. (v) Histogram based techniques are relatively simple to implement, but a key shortcoming of such techniques for multivariate data is that they are not able to capture interactions among the attributes

A comparison of a few statistical network anomaly detection methods is given in Table 3.2

Table 3.2: Comparison of statistical network anomaly detection methods

Author (s)	*Year of publi-cation*	*No of param-eters*	*w*	*x*	*y*	*Data types*	*Dataset used*	*z*	*Detection method*
Eskin [127]	2000	2	O	N	P	Numeric	DARPA99	C_4	Probability Model
Manikopoulos and Papavassiliou [129]	2002	3	D	N	P	Numeric	Real-life	C_2, C_5	Neural Network
Mahoney and Chan [131]	2003	2	C	N	P	-	DARPA99	C_1	LERAD algorithm
Chan et al [130]	2003	2	C	N	P	Numeric	DARPA99	C_1	Learning Rules
Wang and Stolfo [132]	2004	3	C	N	P	Numeric	DARPA99	C_1	Payload based algorithm
Song et al [133]	2007	3	C	N	P	Numeric	KDDcup99	Synthetic intrusive pattern	Gaussian Mixture Model
Chhabra et al [134]	2008	2	D	N	P	Numeric	Real time	C_6	FDR method
Lu and Ghorbani [135]	2009	3	C	N	P,F	Numeric	DARPA99	C_1	Wavelet Analysis
Wattenberg et al [136]	2011	4	C	N	P	Numeric	Real-time	C_2	GLRT Model
Yu [137]	2012	1	C	N	P	Numeric	Real-time	C_2	Adaptive CUSUM

w-indicates centralized (C) or distributed (D) or others (O)
x-the nature of detection as real time (R) or non-real time (N)
y-characterizes packet-based (P) or flow-based (F) or hybrid (H) or others (O)
z-represents the list of attacks handled C_1-all attacks, C_2-denial of service C_3-probe, C_4-user to root, C_5-remote to local and C_6-anomalous

3.2.2 Classification-based Methods and Systems

Classification is the problem of identifying which of a set of categories a new observation belongs to, on the basis of a training set of data containing observations whose category membership is known. Assuming we have two classes whose instances are shown as + and −, and each object can be defined in terms of two attributes or features x_1 and x_2, linear classification tries to find a line between the classes as shown in Figure 3.4(a). The classification boundary may be non-linear

as in Figure 3.4(b). In intrusion detection, the data is high dimensional, not just two. The attributes are usually mixed, numeric and categorial as discussed earlier.

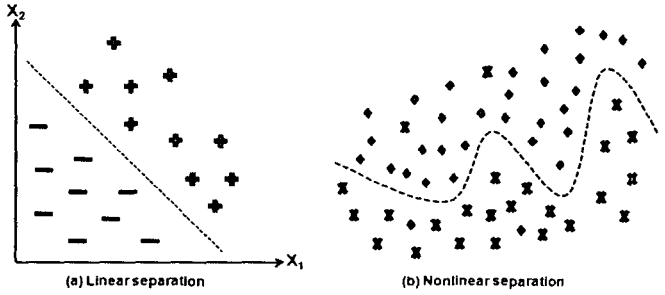


Figure 3.4: Linear and nonlinear classification in 2-D

Thus, classification techniques are based on establishing an explicit or implicit model that enables categorization of network traffic patterns into several classes [143–148]. A singular characteristic of these techniques is that they need labeled data to train the behavioral model, a procedure that places high demands on resources [149]. In many cases, the applicability of machine learning principles such as classification coincides with that of statistical techniques, although the former technique is focused on building a model that improves its performance on the basis of previous results [116]. Several classification based techniques (e.g., k-nearest neighbor, support vector machines, and decision trees) have been applied to anomaly detection in network traffic data.

An example of classification based IDS is Automated Data Analysis and Mining (ADAM) [124] that provides a testbed for detecting anomalous instances. An architecture diagram of ADAM is shown in Figure 3.5. ADAM exploits a combination of classification techniques and association rule mining to discover attacks in a tcpdump audit trail. First, ADAM builds a repository of "normal" frequent itemsets from attack-free periods. Second, ADAM runs a sliding-window based on-line algorithm that finds frequent itemsets in the connections and compares them with those stored in the normal itemset repository, discarding those that are deemed normal. ADAM uses a classifier which has been trained to classify suspicious connections as either a known type of attack or an unknown type or a false alarm.

A few classification-based network anomaly detection methods and NIDSs are described below in brief.

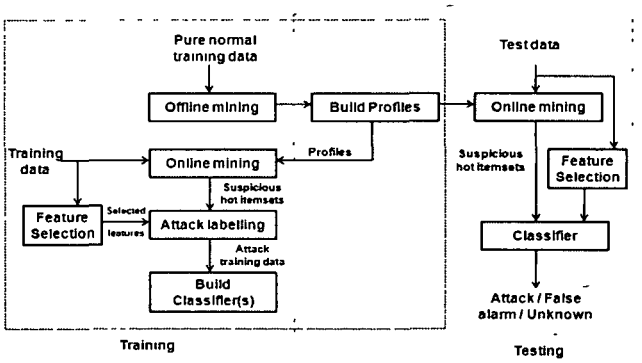


Figure 3.5: Architecture of ADAM system

Abbes et al [150] introduce an approach that uses decision trees with protocol analysis for effective intrusion detection. They construct an adaptive decision tree for each application layer protocol. Detection of anomalies classifies data records into two classes benign and anomalies. The anomalies include a large variety of types such as DoS, scans, and Botnets. Thus, multi-class classifiers are a natural choice, but like any classifier they require expensive hand-labeled datasets and are also not able to identify unknown attacks. Wagner et al. [151] use *one-class classifiers* that can detect new anomalies, i.e., data points that do not belong to the learned class. In particular, they use a one-class SVM classifier proposed by Schölkopf et al. [152]. In such a classifier, the training data is presumed to belong to only one class, and the learning goal during training is to determine a function which is positive when applied to points on the circumscribed boundary around the training points and negative outside. This is also called *semi-supervised classification*. Such an SVM classifier can be used to identify outliers and anomalies. The authors develop a special kernel function that projects data points to a higher dimension before classification. Their kernel function takes into consideration properties of NetFlow data and enables determination of similarity between two windows of IP flow records. They obtain 92% accuracy on average for all attacks classes.

Classification-based anomaly detection methods can usually give better results than unsupervised methods (e.g, clustering-based) because of the use of labeled training examples In traditional classification, new information can be incorporated by retraining with the entire dataset. However, this is time-consuming. Incremental classification algorithms [153] make such training more efficiently. Although

classification-based methods are popular, they cannot detect or predict unknown attack or event until relevant training information is fed for retraining

For a comparison of several classification-based network anomaly detection methods, see Table 3 3.

Table 3.3: Comparison of classification-based network anomaly detection methods

Author (s)	*Year of publication*	*No of parameters*	*w*	*x*	*y*	*Data types*	*Dataset used*	*z*	*Detection method*
Tong et al [143]	2005	4	O	N	P	Numeric	DARPA99, TCPSTAT	C_1	KPCC model
Gaddam et al [144]	2007	3	C	N	P	Numeric	NAD, DED, MSD	C_1	k-means+ID3
Khan et al [154]	2007	3	C	N	P	Numeric	DARPA98	C_1	DGSOT + SVM
Das et al [145]	2008	3	O	N	P	Categorical	KDDcup99	C_1	APD Algorithm
Lu and Tong [146]	2009	2	O	N	P	Numeric	DARPA99	C_1	CUSUM-EM
Qadeer et al [147]	2010	-	C	R	P	-	Real time	C_2	Packet analysis tool
Wagner et al [151]	2011	2	C	R	F	Numeric	Flow Traces	C_2	Kernel OCSVM
Muda et al [155]	2011	2	O	N	O	Numeric	KDDcup99	C_1	KMNB algorithm
Kang et al [148]	2012	2	O	N	P	Numeric	DARPA98	C_1	Differentiated SVDD

w-indicates centralized (C) or distributed (D) or others (O)
x-the nature of detection as real time (R) or non-real time (N)
y-characterizes packet-based (P) or flow-based (F) or hybrid (H) or others (O)
z-represents the list of attacks handled C_1-all attacks, C_2-denial of service, C_3-probe, C_4-user to root, and C_5-remote to local

Several authors have used a combination of classifiers and clustering for network intrusion detection leveraging the advantages of the two methods For example, Gaddam et al. [144] present a method to detect anomalous activities based on a combined approach that uses the k-means clustering algorithm and the ID3 algorithm for decision tree learning [156]. In addition to descriptive features, each data instance includes a label saying whether the instance is normal or anomalous. The first stage of the algorithm partitions the training data into k clusters using Euclidean distance similarity. Obviously, the clustering algorithm does not consider the labels on instances The second stage of the algorithm builds a decision tree on the instances in a cluster. It does so for each cluster so that k separate decision trees are built. The purpose of building decision trees is to overcome two problems that k-means faces: a) *forced assignment*: if the value of k is lower than the number of natural groups, dissimilar instances are forced into the same cluster, and b) *class dominance*, which arises when a cluster contains a large number of instances from

one class, and fewer numbers of instances from other classes The hypothesis is that a decision tree trained on each cluster learns the sub groupings (if any) present within each cluster by partitioning the instances over the feature space To obtain a final decision on classification of a test instance, the decisions of the k-means and ID3 algorithms are combined using two rules (a) the nearest-neighbor rule and (b) the nearest-consensus rule The authors claim that the detection accuracy of the k-means+ID3 method is very high with an extremely low false positive rate on network anomaly data

Support vector machines (SVMs) are very successful maximum margin linear classifiers [157] However, SVMs take a long time for training when the dataset is very large Khan et al [154] reduce the training time for SVMs when classifying large intrusion datasets by using a hierarchical clustering method called Dynamically Growing Self-Organizing Tree (DGSOT) intertwined with the SVMs DGSOT, which is based on artificial neural networks is used to find the boundary points between two classes The boundary points are the most qualified points to train SVMs An SVM computes the maximal margins separating the two classes of data points Only points closest to the margins, called support vectors affect the computation of these margins Other points can be discarded without affecting the final results Khan et al approximate support vectors by using DGSOT They use clustering in parallel with the training of SVMs, without waiting till the end of the building of the tree to start training the SVM The authors find that their approach significantly improves training time for the SVMs without sacrificing generalization accuracy, in the context of network anomaly detection

In addition to the several detection methods viz noted above, we also discuss a classification based IDS known as dependable network intrusion detection system (DNIDS) [158] This IDS is developed based on the Combined Strangeness and Isolation measure of the k-Nearest Neighbor (CSI-KNN) algorithm DNIDS can effectively detect network intrusion while providing continued service under attack The intrusion detection algorithm analyzes characteristics of network data by employing two measures strangeness and isolation These measures are used by a correlation unit to raise intrusion alert along with the confidence information For faster information, DNIDS exploits multiple CSF-KNN classifiers in parallel It

also includes a intrusion tolerant mechanism to monitor the hosts and the classifiers running on them, so that failure of any component can be handled carefully. Sensors capture network packets from a network segment and transform them into connection-based vectors The Detector is a collection of CSI-KNN classifiers that analyze the vectors supplied by the sensors The Manager, Alert Agents, and Maintenance Agents are designed for intrusion tolerance and are installed on a secure administrative server called Station The Manager executes the tasks of generating mobile agents and dispatching them for task execution

Some advantages of classification based anomaly detection methods are the following (i) These techniques are flexible for training and testing They are capable of updating their execution strategies with the incorporation of new information. Hence, adaptability is possible. (ii) They have a high detection rate for known attacks subject to appropriate threshold setting.

Though such methods are popular they have the disadvantages including the following. (i) The techniques are highly dependent on the assumptions made by the classifiers (ii) They consume more resources than other techniques. (iii) They cannot detect or predict unknown attack or event until relevant training information is fed.

3.2.3 Clustering and Outlier-based Methods and Systems

Clustering is the task of assigning a set of objects into groups called *clusters* so that the objects in the same cluster are more similar in some sense to each other than to those in other clusters. Clustering is used in explorative data mining. For example, if we have a set of unlabeled objects in two dimensions, we may be able to cluster them into 5 clusters by drawing circles or ellipses around them, as in Figure 3.6(a). Outliers are those points in a dataset that are highly unlikely to occur given a model of the data, as in Figure 3 6(b) Examples of outliers in a simple dataset are seen in [159]. Clustering and outlier finding are examples of unsupervised machine learning.

Clustering can be performed in network anomaly detection in an offline environment. Such an approach adds additional depth to the administrators' defenses, and

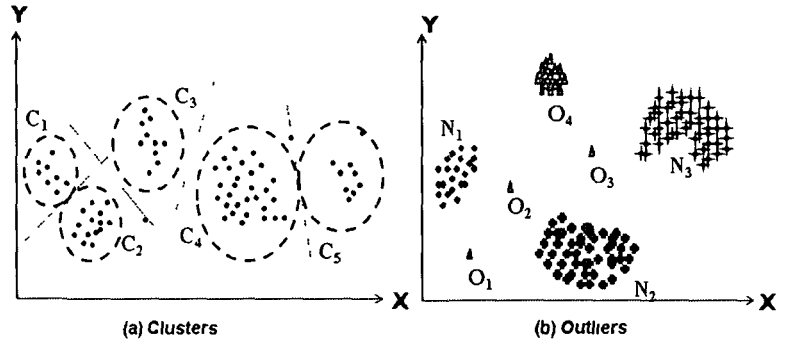


Figure 3.6: Clustering and outliers in 2-D, where C_is are clusters in (a) and O_is are outliers in (b)

allows them to more accurately determine threats against their network through the use of multiple methods on data from multiple sources. Hence, the extensive amount of activities that may be needed to detect intrusion near real time in an online NIDS may be obviated, achieving efficiency [160].

For example, Minnesota INtrusion Detection System (MINDS) [53] is a data mining based system for detecting network intrusions. The architecture of MINDS is given in Figure 3.7 It accepts NetFlow data collected through flow tools as input. Flow tools only capture packet header information and build one-way sessions of flows. The analyst uses MINDS to analyze these data files in batch mode. The reason for running the system in batch mode is not due to the time it takes to analyze these files, but because it is convenient for the analyst to do so Before data is fed into the anomaly detection module, a data filtering step is executed to remove network traffic in which the analyst is not interested.

The first step of MINDS is to extract important features that are used. Then, it summarizes the features based on time windows. After the feature construction step, the known attack detection module is used to detect network connections that correspond to attacks for which signatures are available, and to remove them from further analysis. Next, an outlier technique is activated to assign an anomaly score to each network connection. A human analyst then looks at only the most anomalous connections to determine if they are actual attacks or represent other interesting behavior. The association pattern analysis module of this system is dedicated to summarize the network connections as per the assigned anomaly rank.

The analyst provides feedback after analyzing the summaries created and decides whether these summaries are helpful in creating new rules that may be used in known attack detection

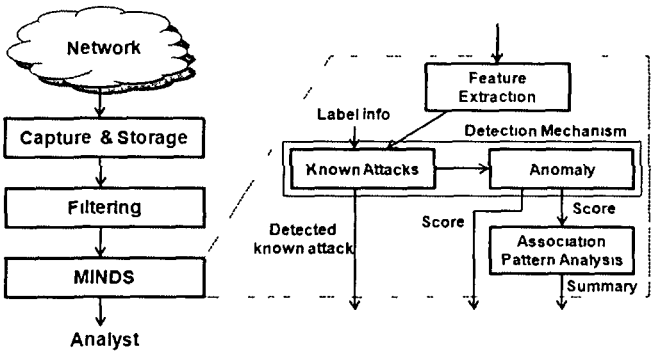


Figure 3.7: Architecture of MINDS system

Clustering techniques are frequently used in anomaly detection These include single-link clustering algorithms k-means (squared error clustering), and hierarchical clustering algorithms to mention a few [31, 63, 161–164]

Sequeira and Zaki [165] present an anomaly based intrusion detection system known as ADMIT that detects intruders by creating user profiles It keeps track of the sequence of commands a user uses as he/she uses a computer A user profile is represented by clustering the sequences of the user's commands The data collection and processing are thus host based The system clusters a user s command sequence using longest common subsequence (LCS) as the similarity metric It uses a dynamic clustering algorithm that creates an initial set of clusters and then refines them by splitting and merging as necessary When a new user types a sequence of commands, it compares the sequence to profiles of users it already has If it is a long sequence, it is broken up to a number of smaller sequences A sequence that is not similar to a normal user's profile is considered anomalous One anomalous sequence is tolerated as noise, but a sequence of anomalous sequences typed by one single user causes the user to be marked as masquerader or a concept drift The system can also use incremental clustering to detect masqueraders Zhang et al [163] report a distributed intrusion detection algorithm that clusters the data twice The first clustering chooses candidate anomalies at Agent IDSs, which are placed in a distributed manner in a network and a second clustering computation

attempts to identify true attacks at the central IDS. The first clustering algorithm is essentially the same as the one proposed by [166]. At each agent IDS, small clusters are assumed to contain anomalies and all small clusters are merged to form a single candidate cluster containing all anomalies The candidate anomalies from various Agent IDSs are sent to the central IDS, which clusters again using a simple single-link hierarchical clustering algorithm. It chooses the smallest k clusters as containing true anomalies. They obtain 90% attacks detection rate on test intrusion data.

Worms are often intelligent enough to hide their activities and evade detection by IDSs. Zhuang et al. [167] propose a method called PAIDS (Proximity-Assisted IDS) to identify the new worms as they begin to spread. PAIDS works differently from other IDSs and has been designed to work collaboratively with existing IDSs such as an anomaly based IDS for enhanced performance. The goal of the designers of PAIDS is to identify new and intelligent fast-propagating worms and thwart their spread, particularly as the worm is just beginning to spread. Neither signature-based nor anomaly based techniques can achieve such capabilities. Zhuang et al.'s approach is based mainly on the observation that during the starting phase of a new worm, the infected hosts are clustered in terms of geography, IP addresses and maybe, even DNSs used.

Some advantages of using clustering are the following. (i) For a partitioning approach, if k can be provided accurately, the task is easy. (ii) Incremental clustering (in supervised mode) techniques are effective for fast response generation. (iii) It is advantageous in case of large datasets to group into similar number of classes for detecting network anomalies, because it reduces the computational complexity during intrusion detection. (iv) It provides a stable performance in comparison to classifiers or statistical methods.

Drawbacks of clustering based methods include the following. (i) Most techniques have been proposed to handle continuous attributes only. (ii) In clustering based intrusion detection techniques, an assumption is that the larger clusters are normal and smaller clusters are attack or intrusion [64]. Without this assumption, it is difficult to evaluate the technique. (iii) Use of an inappropriate proximity measure affects the detection rate negatively (iv) Dynamic updation of profiles is time

consuming.

Several outlier based network anomaly identification techniques are available in [115]. When we use outlier based algorithms, the assumption is that anomalies are uncommon events in a network. Intrusion datasets usually contain mixed, numeric and categorial attributes. Many early outlier detection algorithms worked with continuous attributes only; they ignored categorial attributes or modeled them in manners that caused considerable loss of information.

To overcome this problem, Otey et al. [168] develop a distance measure for data containing a mix of categorical and continuous attributes and use it for outlier based anomaly detection. They define an anomaly score which can be used to identify outliers in the mixed attribute space by considering dependencies among attributes of different types. Their anomaly score function is based on a global model of the data that can be easily constructed by combining local models built independently at each node. They develop an efficient one-pass approximation algorithm for anomaly detection that works efficiently in distributed detection environments with very little loss of detection accuracy. Each node computes its own outliers and the inter-node communication needed to compute global outliers is not significant. In addition, the authors show that their approach works well in dynamic network traffic situations where data, in addition to being streaming, also changes in nature as time progresses leading to concept drift.

Some advantages of outlier-based anomaly detection are the following. (i) It is easy to detect outliers when the datasets are smaller in size. (ii) Bursty and isolated attacks can be identified efficiently using this method.

Drawbacks of outlier-based anomaly detection include the following. (i) Most techniques use both clustering and outlier detection. In such cases the complexity may be high in comparison to other techniques. (ii) The techniques are highly parameter dependent.

A comparison of a few clustering and outlier-based network anomaly detection methods is given in Table 3.4.

Table 3.4: Comparison of clustering and outlier-based network anomaly detection methods

Author (s)	*Year of publication*	*No of parameters*	*w*	*x*	*y*	*Data types*	*Dataset used*	*z*	*Detection method*
Sequeira and Zaki [165]	2002	4	C	R	P	Numeric, Categorical	Real life	Synthetic intrusions	ADMIT
Zhang et al [163]	2005	2	D	N	P	Numeric	KDDcup99	C_1	Cluster-based DIDS
Leung and Leckie [164]	2005	3	C	N	P	Numeric	KDDcup99	C_1	fpMAFIA algorithm
Otey et al [168]	2006	5	C	N	P	Mixed	KDDcup99	C_1	FDOD algorithm
Jiang et al [7]	2006	3	C	N	P	Mixed	KDDcup99	C_1	CBUID algorithm
Chen and Chen [169]	2008	-	O	N	-	-	-	C_3	AAWP model
Zhang et al [63]	2009	2	O	N	P	Mixed	KDDcup99	C_1	KD algorithm
Zhuang et al [167]	2010	2	R	C	P	-	Real time	C_6	PAIDS model
Casas et al [31]	2012	2	N	C	F	Numeric	KDDcup99 Real time	C_1	UNIDS method

w-indicates centralized (C) or distributed (D) or others (O)
x-the nature of detection as real time (R) or non-real time (N)
y-characterizes packet-based (P) or flow-based (F) or hybrid (H) or others (O)
z-represents the list of attacks handled C_1-all attacks C_2-denial of service C_3-probe, C_4-user to root, C_5-remote to local and C_6-worms

3.2.4 Soft Computing-based Methods and Systems

Soft computing techniques are suitable for network anomaly detection because often one cannot find exact solutions. Soft computing is usually thought of as encompassing methods such as genetic algorithms, artificial neural networks, fuzzy sets, and rough sets. We describe several soft computing methods and systems for network anomaly detection below.

Genetic Algorithm Approaches

Genetic algorithms are population-based adaptive heuristic search techniques based on evolutionary ideas. The approach begins with conversion of a problem into a framework that uses a chromosome like data structure. Balajinath and Raghavan [170] present a genetic intrusion detector (GBID) based on learning of individual user behavior. User behavior is described as 3-tuple <matching index, entropy index, newness index> and is learnt using a genetic algorithm. This behavior profile is used to detect intrusion based on past behavior. Khan [171] uses genetic algorithms to develop rules for network intrusion detection. A chromosome in an

individual contains genes corresponding to attributes such as the service, flags, logged in or not, and super-user attempts Khan concludes that attacks that are common can be detected more accurately compared to uncommon attributes

Artificial Neural Networks Approaches

Artificial neural networks (ANN) are motivated by the recognition that the human brain computes in an entirely different way from the conventional digital computer [172] The brain organizes its constituents, known as neurons, so as to perform certain computations (e g , pattern recognition, perception, and motor control) many times faster than the fastest digital computer To achieve good performance, real neural networks employ massive interconnections of neurons Neural networks acquire knowledge of the environment through a process of learning, which systematically changes the interconnection strengths or synaptic weights of the network to attain a desired design objective

An example of ANN-based IDS is RT-UNNID [173] This system is capable of intelligent real time intrusion detection using unsupervised neural networks (UNN) The architecture of RT-UNNID is given in Figure 3 8 The first module captures and preprocesses the real time network traffic data for the protocols TCP, UDP and ICMP It also extracts the numeric features and converts them into binary or normalized form The converted data is sent to the UNN based detection engine that uses adaptive resonance theory (ART) and self-organizing map (SOM) [174, 175] neural networks Finally, the output of the detection engine is sent to the responder for recording in the user's system log file and to generate alarm when detecting attacks RT-UNNID can work in real time to detect known and unknown attacks in network traffic with high detection rate

Cannady's approach [176] autonomously learns new attacks rapidly using modified reinforcement learning His approach uses feedback for signature update when a new attack is encountered and achieves satisfactory results An improved approach to detect network anomalies using a hierarchy of neural networks is introduced in [177] The neural networks are trained using data that spans the entire normal space and are able to recognize unknown attacks effectively Liu et al [178] report a real time solution to detect known and new attacks in network traffic using

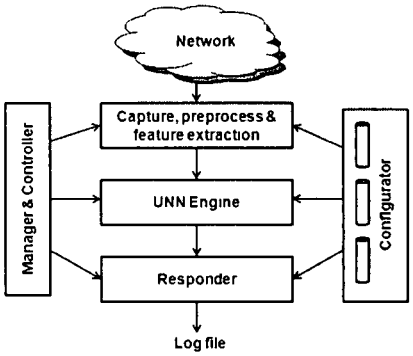


Figure 3.8: Architecture of RT-UNNID system

unsupervised neural nets. It uses a hierarchical intrusion detection model using principal components analysis (PCA) neural networks to overcome the shortcomings of single-level structures. However, Sun et al. [179] present a wavelet neural network (WNN) based intrusion detection method. It reduces the number of the wavelet basic functions by analyzing the sparseness property of sample data to optimize the wavelet network to a large extent. The learning algorithm trains the network using gradient descent.

In addition to the detection methods, we discuss a few IDSs below.

Network self-organizing maps (NSOM) [180] is a network IDS developed using self-organizing maps (SOM). It detects anomalies by quantifying the usual or acceptable behavior and flags irregular behavior as potentially intrusive. To classify real time traffic, it uses a structured SOM. It continuously collects network data from a network port, preprocesses that data and selects the features necessary for classification. Then it starts the classification process a chunk of packets at a time and then sends the resulting classification to a graphical tool that portrays the activities that are taking place on the network port dynamically as it receives more packets. The hypothesis is that routine traffic that represents normal behavior would be clustered around one or more cluster centers and any irregular traffic representing abnormal and possibly suspicious behavior would be clustered in addition to the normal traffic clustering. The system is capable of classifying regular vs. irregular and possibly intrusive network traffic for a given host.

POSEIDON (PAYL Over SOM for Intrusion DetectiON) [181] is a two-tier network intrusion detection system. The first tier consists of a self-organizing map

(SOM), and is used exclusively to classify payload data. The second tier consists of a light modification of the PAYL system [132]. Tests using the DARPA99 dataset show a higher detection rate and lower number of false positives than PAYL and PHAD [182]

Fuzzy Set Theoretic Approaches

Fuzzy network intrusion detection systems exploit fuzzy rules to determine the likelihood of specific or general network attacks [183,184]. A fuzzy input set can be defined for traffic in a specific network.

Tajbakhsh et al. [185] describe a novel method for building classifiers using fuzzy association rules and use it for network intrusion detection. The fuzzy association rule sets are used to describe different classes· normal and anomalous. Such fuzzy association rules are *class association rules* where the consequents are specified classes. Whether a training instance belongs to a specific class is determined by using matching metrics proposed by the authors. The fuzzy association rules are induced using normal training samples. A test sample is classified as normal if the compatibility of the rule set generated is above a certain threshold; those with lower compatibility are considered anomalous. The authors also propose a new method to speed up the rule induction algorithm by reducing items from extracted rules.

Mabu et al. report a novel fuzzy class-association-rule mining method based on genetic network programming (GNP) for detecting network intrusions [186]. GNP is an evolutionary optimization technique, which uses directed graph structures instead of strings in standard genetic algorithms leading to enhanced representation ability with compact descriptions derived from possible node reusability in a graph. Xian et al. [187] present a novel unsupervised fuzzy clustering method based on clonal selection for anomaly detection. The method is able to obtain global optimal clusters more quickly than competing algorithms with greater accuracy.

In addition to the fuzzy set theoretic detection methods, we discuss two IDSs, viz., NFIDS and FIRE below.

NFIDS [188] is a neuro-fuzzy anomaly-based network intrusion detection system. It comprises three tiers. Tier-I contains several Intrusion Detection Agents

(IDAs) IDAs are IDS components that monitor the activities of a host or a network and report the abnormal behavior to Tier-II. Tier-II agents detect the network status of a LAN based on the network traffic that they observe as well as the reports from the Tier-I agents within the LAN. Tier-III combines higher-level reports, correlates data, and sends alarms to the user interface There are four main types of agents in this system: TCPAgent, which monitors TCP connections between hosts and on the network, UDPAgent, which looks for unusual traffic involving UDP data, ICMPAgent, which monitors ICMP traffic and PortAgent, which looks for unusual services in the network

Fuzzy intrusion recognition engine (FIRE) [183] is an anomaly-based intrusion detection system that uses fuzzy logic to assess whether malicious activity is taking place on a network. The system combines simple network traffic metrics with fuzzy rules to determine the likelihood of specific or general network attacks. Once the metrics are available, they are evaluated using a fuzzy set theoretic approach. The system takes on fuzzy network traffic profiles as inputs to its rule set and report maliciousness.

Rough Set Approaches

A rough set is an approximation of a crisp set (i.e., a regular set) in terms of a pair of sets that are its lower and upper approximations In the standard and original version of rough set theory [189], the two approximations are crisp sets, but in other variations the approximating sets may be fuzzy sets. The mathematical framework of rough set theory enables modeling of relationships with a minimum number of rules.

Rough sets have two useful features [190]: (i) enabling learning with small size training datasets (ii) and overall simplicity. They can be applied to anomaly detection by modeling normal behavior in network traffic. For example, in [191], the authors present a fuzzy rough C-means clustering technique for network intrusion detection by integrating fuzzy set theory and rough set theory to achieve high detection rate. Chen et al present a two-step classifier for network intrusion detection [192]. Initially, it uses rough set theory for feature reduction and then a support vector machine classifier for final classification. They obtain 89% accuracy

on network anomaly data.

Advantages of soft computing based anomaly detection methods include the following. (i) Such learning systems detect or categorize persistent features without any feedback from the environment. (ii) Due to the adaptive nature of ANNs, it is possible to train and test instances incrementally using certain algorithms. Multi-level neural network based techniques are more efficient than single level neural networks. (iii) Unsupervised learning using competitive neural networks is effective in data clustering, feature extraction and similarity detection. (iv) Rough sets are useful in resolving inconsistency in the dataset and to generate a minimal, non-redundant and consistent rule set

Some disadvantages of soft computing methods are the following. (i) Over-fitting may happen during neural network training (ii) If a credible amount of normal traffic data is not available, the training of the techniques becomes very difficult. (iii) Most methods have scalability problems. (iv) Rough set based rule generation suffers from proof of completeness. (v) In fuzzy association rule based techniques, reduced, relevant rule subset identification and dynamic rule updation at runtime is a difficult task.

Table 3.5 gives a comparison of several soft computing-based anomaly detection methods.

3.2.5 Knowledge-based Methods and Systems

In knowledge-based methods, network or host events are checked against predefined rules or patterns of attack The goal is to represent the known attacks in a generalized fashion so that handling of actual occurrences become easier. Examples of knowledge-based methods are expert systems, rule based, ontology based, logic based and state-transition analysis [195–198]

These techniques search for instances of known attacks. by attempting to match traffic patterns with pre-determined attack representations. The search begins like other intrusion detection techniques, with a complete lack of knowledge. Subsequent matching of activities against a known attack helps acquire knowledge and enter into a region with higher confidence. Finally, it can be shown that an event or

Table 3.5: Comparison of soft computing-based network anomaly detection methods

Author (s)	*Year of publication*	*No of parameters*	*w*	*x*	*y*	*Data types*	*Dataset used*	*z*	*Detection method*
Cannady [176]	2000	2	O	N	P	Numeric	Real-life	C_2	CMAC-based model
Balajinath and Raghavan [170]	2001	3	O	N	O	Categorical	User command	C_4	Behavior Model
Lee and Heinbuch [177]	2001	3	C	N	P	-	Simulated data	C_2	TNNID model
Xian et al [187]	2005	3	C	N	P	Numeric	KDDcup99	C_1	Fuzzy K-means
Amini et al [173]	2006	2	C	R	P	Categorical	KDDcup99, Real-life	C_1	RT-UNNID system
Chimphlee et al [191]	2006	3	C	N	P	Numeric	KDDcup99	C_1	Fuzzy Rough C-means
Liu et al [178]	2007	2	C	N	P	Numeric	KDDcup99	C_1	HPCANN Model
Adetunmbi et al [193]	2008	2	C	N	P	Numeric	KDDcup99	C_1	LEM2 and K-NN
Chen et al [192]	2009	3	C	N	P	Numeric	DARPA98	C_2	RST-SVM technique
Mabu et al [186]	2011	3	C	N	P	Numeric	KDDcup99	C_1	Fuzzy-ARM based on GNP
Visconti and Tahayori [194]	2011	2	O	N	P	Numeric	Real-life	C_2	Interval type-2 fuzzy set
Geramiraz et al [184]	2012	2	O	N	P	Numeric	KDDcup99	C_1	Fuzzy rule based model

w-indicates centralized (C) or distributed (D) or others (O)
x-the nature of detection as real time (R) or non-real time (N)
y-characterizes packet-based (P) or flow-based (F) or hybrid (H) or others (O)
z-represents the list of attacks handled C_1-all attacks, C_2-denial of service C_3-probe, C_4-user to root and C_5-remote to local

activity has reached maximum anomaly score.

An example knowledge-based system is state transition analysis tool (STAT) [199]. Its architecture is given in Figure 3.9 It models traffic data as a series of state changes that lead from secure state to a target compromised state STAT is composed of three main components: knowledge base, inference engine and decision engine. The audit data preprocessor reformats the raw audit data to send as input to the inference engine. The inference engine monitors the state transitions extracted from the preprocessed audit data and then compares these states with the states available within the knowledge base. The decision engine monitors the improvement of the inference engine for matching accuracy of the state transitions. It also specifies the action(s) to be taken based on results of the inference engine and the decision table. Finally, the decision results are sent to the SSO (Site Security Officer) interface for action. It can detect cooperative attackers and attacks across user sessions well.

A few prominent knowledge-based network anomaly detection methods and

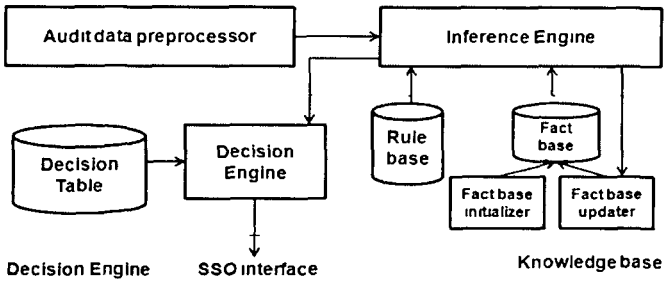


Figure 3.9: Architecture of STAT system

NIDSs are given below

Rule-based and Expert System-based Approaches

The expert system approach is one of the most widely used knowledge-based methods [200 201] An expert system, in the traditional sense, is a rule-based system, with or without an associated knowledge base An expert system has a rule engine that matches rules against the current state of the system, and depending on the results of matching, fires one or more rules

Snort [161] is a quintessentially popular rule-based IDS This open-source IDS matches each packet it observes against a set of rules The antecedent of a Snort rule is a boolean formula composed of predicates that look for specific values of fields present in IP headers transport headers and in the payload Thus, Snort rules identify attack packets based on IP addresses, TCP or UDP port numbers, ICMP codes or types, and contents of strings in the packet payload Snort's rules are arranged into priority classes based on potential impact of alerts that match the rules Snort's rules have evolved over its history of 15 years Each Snort rule has associated documentation with the potential for false positives and negatives, together with corrective actions to be taken when the rule raises an alert Snort rules are simple and easily understandable Users can contribute rules when they observe new types of anomalous or malicious traffic Currently, Snort has over 20,000 rules, inclusive of those submitted by users

An intrusion detection system like Snort can run on a general purpose computer and can try to inspect all packets that go through the network However, monitoring packets comprehensively in a large network is obviously an expensive task since it

requires fast inspection on a large number of network interfaces. Many hundreds of rules may have to be matched concurrently, making scaling almost impossible.

To scale to large networks that collect flow statistics ubiquitously, Duffield et al. [202] use the machine learning algorithm called Adaboost [203] to translate packet level signatures to work with flow level statistics. The algorithm is used to correlate the packet and flow information In particular, the authors associate packet level network alarms with a feature vector they create from flow records on the same traffic. They create a set of rules using flow information with features similar to those used in Snort rules. They also add numerical features such as the number of packets of a specific kind flowing within a certain time period. Duffield et al. train Adaboost on concurrent flow and packet traces. They evaluate the system using real time network traffic data with more than a billion flows over 29 days, and show that their performance is comparable to Snort's with NetFlow data.

Prayote and Compton [204] present an approach to anomaly detection that attempts to address the brittleness problem in which an expert system makes a decision that human common sense would recognize as impossible. They use a technique called *prudence* [205], in which for every rule, the upper and lower bounds of each numerical variable in the data seen by the rule are recorded, as well as a list of values seen for enumerated variables The expert system raises a warning when a new value or a value outside the range is seen in a data instance. They improve the approach by using a simple probabilistic technique to decide if a value is an outlier. When working with network anomaly data, the authors partition the problem space into smaller subspaces of homogeneous traffic, each of which is represented with a separate model in terms of rules The authors find that this approach works reasonably well for new subspaces when little data has been observed They claim 0% false negative rate in addition to very low false positive rate. Scheirer and Chuah [206] report a syntax-based scheme that uses variable-length partition with multiple break marks to detect many polymorphic worms. The prototype is the first NIDS that provides semantics-aware capability, and can capture polymorphic shell codes with additional stack sequences and mathematical operations.

The main advantages of knowledge-based anomaly detection methods include

the following. (i) These techniques are robust and flexible. (ii) These techniques have high detection rate, if a substantial knowledge base can be acquired properly about attacks as well as normal instances.

Some disadvantages of knowledge-based methods are the following. (i) The development of high-quality knowledge is often difficult and time-consuming. (ii) Due to non-availability of biased normal and attack data, such a method may generate a large number of false alarms (iii) Such a method may not be able to detect rare or unknown attacks. (iv) Dynamic updation of rule or knowledge-base is a costly affair.

A comparison of knowledge-based anomaly detection methods is given in Table 3.6.

Table 3.6: Comparison of knowledge based network anomaly detection methods

Author (s)	*Year of publi-cation*	*No of param-eters*	*w*	*x*	*y*	*Data types*	*Dataset used*	*z*	*Detection method*
Noel et al [195]	2002	-	O	N	O	-	-	-	Attack Guilt Model
Sekar et al [196]	2002	3	O	N	P	Numeric	DARPA99	C_1	Specification Based Model
Tapiador et al [207]	2003	3	C	N	P	Numeric	Real-life	C_2	Markov Chain Model
Hung and Liu [208]	2008	-	O	N	P	Numeric	KDDcup99	C_1	Ontology based
Shabtai et al [209]	2010	2	O	N	O	-	Real-life	C_2	Incremental KBTA

w-indicates centralized (C) or distributed (D) or others (O)
x-the nature of detection as real time (R) or non-real time (N)
y-characterizes packet-based (P) or flow-based (F) or hybrid (H) or others (O)
z-represents the list of attacks handled C_1-all attacks, C_2-denial of service, C_3-probe C_4-user to root, and C_5-remote to local

3.2.6 Methods and Systems based on Combination Learners

In this section, we present a few methods and systems which use combinations of multiple techniques, usually classifiers.

Ensemble-based Methods and Systems

The idea behind the ensemble methodology is to weigh several individual classifiers, and combine them to obtain an overall classifier that outperforms every one of them [210–214]. These techniques weigh the individual opinions, and combine

them to reach a final decision. Ensemble-based methods are categorized based on the approaches used. Three main approaches to develop ensembles are (i) bagging, (ii) boosting, and (iii) stack generalization. *Bagging* (Bootstrap Aggregating) increases classification accuracy by creating an improved composite classifier into a single prediction by combining the outputs of learnt classifiers. *Boosting* builds an ensemble incrementally by training mis-classified instances obtained from the previous model. *Stack generalization* achieves the high generalization accuracy by using output probabilities for every class label from the base-level classifiers.

Octopus-IIDS [215] is an example of ensemble IDS. The architecture of this system is shown in Figure 3.10. It is developed using two types of neural networks (Kohonen and Support Vector Machines). The system is composed of two layers: classifier and anomaly detection. The classifier is responsible for capturing and preprocessing of network traffic data. It classifies the data into four main categories: DoS, probe, U2R and R2L. A specific class of attack is identified in the anomaly detection layer. The authors claim that the IDS works effectively in small scale networks.

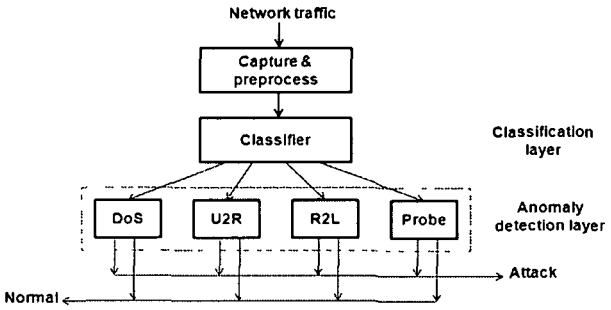


Figure 3.10: Architecture of Octopus-IIDS system

Chebrolu et al. [216] present an ensemble approach by combining two classifiers, Bayesian networks (BN) and Classification and Regression Trees (CART) [138,217]. A hybrid architecture for combining different feature selection algorithms for real world intrusion detection is also incorporated for getting better results. Perdisci et al. [218] construct a high speed payload anomaly IDS using an ensemble of one-class SVM classifiers intended to be accurate and hard to evade.

Folino et al. [219] introduce a distributed data mining algorithm to improve

detection accuracy when classifying malicious or unauthorized network activity using genetic programming (GP) extended with the ensemble paradigm. Their data is distributed across multiple autonomous sites and the learner component acquires useful knowledge from data in a cooperative way and uses network profiles to predict abnormal behavior with better accuracy Nguyen et al [220] build an individual classifier using both the input feature space and an additional subset of features given by k-means clustering. The ensemble combination is calculated based on the classification ability of classifiers on different local data segments given by k-means clustering.

Beyond the above methods, some ensemble based IDSs are given below.

The paradigm of Multiple Classifier System (MCS) has also been used to build misuse detection IDSs. Classifiers trained on different feature subsets are combined to achieve better classification accuracy than the individual classifiers. In such a NIDS, network traffic is serially processed by each classifier. At each stage, a classifier may either decide for one attack class or send the pattern to another stage, which is trained on more difficult cases Reported results show that an MCS improves the performance of IDSs based on statistical pattern recognition techniques. For example, CAMNEP [221] is a fast prototype agent-based NIDS designed for high-speed networks. It integrates several anomaly detection techniques, and operates on a collective trust model within a group of collaborative detection agents. The anomalies are used as input for trust modeling Aggregation is performed by extended trust models of generalized situated identities, represented by a set of observable features. The system is able to perform real time surveillance of gigabit networks.

McPAD (Multiple Classifier Payload-based Anomaly Detector) [222] is an effective payload-based anomaly detection system that consists of an ensemble of one-class classifiers. It is very accurate in detecting network attacks that bear some form of shell-code in the malicious payload This detector performs well even in the case of polymorphic attacks. Furthermore, the authors tested their IDS with advanced polymorphic blending attacks and showed that even in the presence of such sophisticated attacks, it is able to obtain a low false positive rate.

An ensemble method is advantageous because it obtains higher accuracy than

the individual techniques. The major advantages are the following. (i) Even if the individual classifiers are weak, the ensemble methods perform well by combining multiple classifiers. (ii) Ensemble methods can scale for large datasets. (iii) Ensemble classifiers need a set of controlling parameters that are comprehensive and can be easily tuned. (iv) Among existing approaches, Adaboost and Stack generalization are very effective because they can exploit the diversity in predictions by multiple base level classifiers.

Some disadvantages of ensemble based methods include the following. (i) Selecting a subset of consistent performing and unbiased classifiers from a pool of classifiers is difficult. (ii) The greedy approach for selecting sample datasets is slow for large datasets. (iii) It is difficult to obtain real time performance.

A comparison of ensemble based network anomaly detection methods is given in Table 3.7.

Table 3.7: Comparison of ensemble-based network anomaly detection methods

Author (s)	*Year of publication*	*Combination strategy*	*w*	*x*	*y*	*Data types*	*Dataset used*	*z*	*Detection method*
Chebrolu et al [216]	2005	Weighted voting	O	N	P	Numeric	KDDcup99	C_1	Class specific ensemble model
Perdisci et al [218]	2006	Majority voting	O	N	Pay	-	Operational points	Synthetic intrusions	One-class classifier model
Borji [211]	2007	Majority voting	O	N	P	Numeric	DARPA98	C_1	Heterogeneous classifiers model
Perdisci et al [222]	2009	Min and Max probability	O	R	Pay	-	DARPA98	C_1	McPAD model
Folino et al [219]	2010	Weighted majority voting	O	N	P	Numeric	KDDcup99	C_1	GEdIDS model
Noto et al [214]	2010	Information theoretic	O	N	-	Numeric	UCI	None	FRaC model
Nguyen et al [220]	2011	Majority voting	O	N	P	Numeric	KDDcup99	C_1	Cluster based ensemble
Khreich et al [25]	2012	Learn and combine	O	N	pay	Numeric	UNM	C_4	EoHMMs based

w-indicates centralized (C) or distributed (D) or others (O)
x-the nature of detection as real time (R) or non-real time (N)
y-characterizes packet-based (P) or flow-based (F) or payload-based (pay) or hybrid (H) or others (O)
z-represents the list of attacks handled C_1-all attacks C_2-denial of service. C_3-probe, C_4-user to root, and C_5-remote to local

Fusion-based Methods and System

With an evolving need of automated decision making it is important to improve classification accuracy compared to the stand-alone general decision based techniques even though such a system may have several disparate data sources So, a suitable combination of these is known as fusion approach Several fusion-based techniques have been applied to network anomaly detection [223–227] A classification of such techniques is as follows (i) data level, (ii) feature level, and (iii) decision level Some methods only address the issue of operating in a space of high dimensionality with features divided into semantic groups Others attempt to combine classifiers trained on different features divided based on hierarchical abstraction levels or the type of information contained

Giacinto et al [223] provide a pattern recognition approach to network intrusion detection employing a fusion of multiple classifiers Five different decision fusion methods are assessed by experiments and their performances compared Shifflet [224] discusses a platform that enables a multitude of techniques to work together towards creating a more realistic fusion model of the state of a network, able to detect malicious activity effectively A heterogenous data level fusion for network anomaly detection is added by Chatzigiannakis et al [228] They use the Demster-Shafer Theory of Evidence and Principal Components Analysis for developing the technique

dLEARNIN [225] is an ensemble of classifiers that combines information from multiple sources It is explicitly tuned to minimize the cost of errors dLEARNIN is shown to achieve state-of-the-art performance, better than competing algorithms The cost minimization strategy dCMS attempts to minimize the cost to a significant level Gong et al [229] contribute a neural network based data fusion method for intrusion data analysis and pruning to filter information from multiple sensors to get high detection accuracy However, HMMPayl [230] is an example of fusion based IDS, where the payload is represented as a sequence of bytes, and the analysis is performed using Hidden Markov Models (HMM) The algorithm extracts features and uses HMM to guarantee the same expressive power as that of n-gram analysis, while overcoming its computational complexity HMMPayl follows the

Multiple Classifiers System paradigm to provide better classification accuracy, to increase the difficulty of evading the IDS, and to mitigate the weaknesses due to a non-optimal choice of HMM parameters

Some advantages of fusion methods are the following (i) Data fusion is effective in increasing timeliness of attack identification and in reducing false alarm rates. (ii) Decision level fusion with appropriate training data usually yields high detection rate

Some drawbacks are the following. (i) The computational cost is high for rigorous training on the samples. (ii) Feature level fusion is a time consuming task. Also, the biases among the base classifiers affect the fusion process. (iii) Building hypotheses for different classifiers is a difficult task.

A comparison of fusion-based network anomaly detection methods is given in Table 3.8

Table 3.8: Comparison of fusion-based network anomaly detection methods

Author (s)	*Year of publication*	*Fusion level*	*w*	*x*	*y*	*Data types*	*Dataset used*	*z*	*Detection method*
Giacinto et al [223]	2003	Decision	O	N	P	Numeric	KDDcup99	C_1	MCS Model
Shifflet [224]	2005	Data	O	N	O	-	-	None	HSPT algorithm
Chatzigiannakis et al [228]	2007	Data	C	N	P	-	NTUA GRNET	C_2	D-S algorithm
Parikh and Chen [225]	2008	Data	C	N	P	Numeric	KDDcup99	C_1	dLEARNIN system
Gong et al [229]	2010	Data	C	N	P	Numeric	KDDcup99	C_1	IDEA model
Ariu et al [230]	2011	Decision	C	R	Pay	-	DARPA98, real-life	C_1	HMMPayl model
Yan and Shao [227]	2012	Decision	O	N	F	Numeric	Real time	C_2, C_3	EWMA model

w-indicates centralized (C) or distributed (D) or others (O)
x-the nature of detection as real time (R) or non-real time (N)
y-characterizes packet-based (P) or flow-based (F) or payload-based (pay) or hybrid (H) or others (O)
z-represents the list of attacks handled C_1-all attacks, C_2-denial of service C_3-probe, C_4-user to root. and C_5-remote to local

Hybrid Methods and System

Most current network intrusion detection systems employ either misuse detection or anomaly detection However, misuse detection cannot detect unknown intrusions, and anomaly detection usually has high false positive rate [231]. To overcome the limitations of the techniques, hybrid methods are developed by exploiting features from several network anomaly detection approaches [32–34]. Hybridization of sev-

eral methods increases performance of IDSs

For example, RT-MOVICAB-IDS, a hybrid intelligent IDS is introduced in [232] It combines ANN and CBR (case based reasoning) within a Multi-Agent System (MAS) to detect intrusion in dynamic computer networks The dynamic real time multi-agent architecture allows the addition of prediction agents (both reactive and deliberative) In particular, two of the deliberative agents deployed in the system incorporate temporal-bounded CBR This upgraded CBR is based on an anytime approximation, which allows the adaptation of this paradigm to real time requirements

A hybrid approach to provide host security that prevents binary code injection attacks known as the FLIPS (Feedback Learning IPS) model is proposed by [233]. It incorporates three major components an anomaly based classifier, a signature based filtering scheme, and a supervision framework that employs Instruction Set Randomization (ISR) Capturing the injected code allows FLIPS to construct signatures for zero-day exploits Peddabachigari et al [234] present a hybrid approach that combines Decision trees (DT) and SVMs as a hierarchical hybrid intelligent system model (DTSVM) for intrusion detection It maximizes detection accuracy and minimizes computational complexity

Zhang et al [235] propose a systematic framework that applies a data mining algorithm called *random forests* in building a misuse, anomaly, and hybrid-network based IDS The hybrid detection system improves detection performance by combining the advantages of both misuse and anomaly detection Tong et al [236] discuss a hybrid RBF/Elman neural network model that can be employed for both anomaly detection and misuse detection It can detect temporally dispersed and collaborative attacks effectively because of its memory of past events A intelligent hybrid IDS model based on neural networks is introduced by [237] The model is flexible, extended to meet different network environments, improves detection performance and accuracy Selim et al [238] report a hybrid intelligent IDS to improve the detection rate for known and unknown attacks It consists of multiple levels hybrid neural networks and decision trees The technique is evaluated using NSL-KDD dataset and results were promising

Advantages of hybrid methods include the following (i) Such a method exploits

major features from both signature and anomaly based network anomaly detection. (ii) Such methods can handle both known and unknown attacks.

Drawbacks of hybrid methods include the following (i) Lack of appropriate hybridization may lead to high computational cost (ii) Dynamic updation of rule or profile or signature still remains difficult.

Table 3.9 presents a comparison of a few hybrid network anomaly detection methods.

Table 3.9: Comparison of hybrid network anomaly detection methods

Author (s)	*Year of publi-cation*	*No of param-eters*	*w*	*x*	*y*	*Data types*	*Dataset used*	*z*	*Detection method*
Locasto et al [233]	2005	2	C	R	P	-	Real-life	C_2	FLIPS model
Zhang and Zulkernine [32]	2006	2	C	N	P	Numeric	KDDcup99	C_1	Random forest based hybrid algorithm
Peddabachigari et al [234]	2007	2	C	N	P	Numeric	KDDcup99	C_1	DT-SVM hybrid model
Zhang et al [235]	2008	2	C	N	P	Numeric	KDDcup99	C_1	RFIDS model
Aydin et al [33]	2009	3	C	N	P	-	DARPA98 IDEVAL	C_1	Hybrid signature based IDS model
Tong et al [236]	2009	1	C	N	P	Numeric	DARPA-BSM	C_1	Hybrid RBF/Elman NN
Yu [237]	2010	1	C	N	-	-	-	-	Hybrid NIDS
Arumugam et al [231]	2010	-	C	N	P	Numeric	KDDcup99	C_1	Multi-stage hybrid IDS
Selim et al [238]	2011	-	C	N	P	Numeric	KDDcup99	C_1	Hybrid multi-level IDS model
Panda et al [34]	2012	2	C	N	P	Numeric	NSL-KDD KDDcup99	C_1	DTFF and FFNN

w-indicates centralized (C) or distributed (D) or others (O)
x-the nature of detection as real time (R) or non-real time (N)
y-characterizes packet based (P) or flow based (F) or hybrid (H) or others (O)
z-represents the list of attacks handled C_1-all attacks C_2-denial of service C_3-probe, C_4-user to root, and C_5-remote to local

3.2.7 Discussion

After a long and elaborate discussion of many intrusion detection methods and anomaly based network intrusion detection systems under several categories, we make a few observations.

(i) Each class of anomaly based network intrusion detection methods and systems has unique strengths and weaknesses. The suitability of an anomaly detection technique depends on the nature of the problem attempted to address. Hence, providing a single integrated solution to every anomaly detection problem may

not be feasible

(ii) Various methods face various challenges when complex datasets are used The nearest neighbor and clustering techniques suffer when the number of dimensions is high because the distance measures in high dimensions are not able to differentiate well between normal and anomalous instances

Spectral techniques explicitly address the high dimensionality problem by mapping data to a lower dimensional projection But their performance is highly dependent on the assumption that normal instances and anomalies are distinguishable in the projected space A classification technique often performs better in such a scenario However, it requires labeled training data for both normal and attack classes The improper distribution of these training data often makes the task of learning more challenging

Semi-supervised nearest neighbor and clustering techniques that only use normal labels can often be more effective than classification-based techniques In situations where identifying a good distance measure is difficult, classification or statistical techniques may be a better choice However, the success of the statistical techniques is largely influenced by the applicability of the statistical assumptions in the specific real life scenarios

(iii) For real-time intrusion detection, the complexity of the anomaly detection process plays a vital role In case of classification clustering and statistical methods although training is expensive, they are still acceptable because testing is fast and training is offline In contrast, techniques such as nearest neighbor and spectral techniques which do not have a training phase, have an expensive testing phase which can be a limitation in a real setting

(iv) Anomaly detection techniques typically assume that anomalies in data are rare when compared to normal instances Generally, such assumptions are valid but not always Often unsupervised techniques suffer from large false alarm rates when anomalies are in bulk amounts Techniques operating in supervised or semi-supervised modes [239] can be applied to detect bulk anomalies

We also perform a comparison of the anomaly based network intrusion detection systems that we have discussed throughout this chapter based on parameters such as mode of detection (host based, network based or both), detection approach (misuse, anomaly or both), nature of detection (online or offline), nature of processing (centralized or distributed), data gathering mechanism (centralized or distributed) and approach of analysis. A comparison chart is given in Table 3.10.

Table 3.10: Comparison of NIDSs

Name of IDS	*Year of publication*	*a*	*b*	*c*	*d*	*e*	*Approach*
STAT [199]	1995	H	M	R	C	C	Knowledge-based
FIRE [183]	2000	N	A	N	C	C	Fuzzy Logic
ADAM [124]	2001	N	A	R	C	C	Classification
HIDE [125]	2001	N	A	R	C	D	Statistical
NSOM [180]	2002	N	A	R	C	C	Neural network
NFIDS [188]	2003	N	A	N	C	C	Neuro Fuzzy Logic
N@G [141]	2003	Hy	B	R	C	C	Statistical
MINDS [53]	2004	N	A	R	C	C	Clustering and Outlier-based
FSAS [142]	2006	N	A	R	C	C	Statistical
POSEIDON [181]	2006	N	A	R	C	C	SOM & Modified PAYL
RT-UNNID [173]	2006	N	A	R	C	C	Neural Network
DNIDS [158]	2007	N	A	R	C	C	CSI-KNN-based
CAMNEP [221]	2008	N	A	R	C	C	Agent-based Trust and Reputation
McPAD [222]	2009	N	A	N	C	C	Multiple classifier
Octopus-IIDS [215]	2010	N	A	N	C	C	Neural network & SVM
HMMPayl [230]	2011	N	A	R	C	C	HMM model
RT-MOVICAB-IDS [232]	2011	N	A	R	C	C	Hybrid IDS

a-represents the types of detection such as host-based (H) or network-based (N) or hybrid (H)
b-indicates the class of detection mechanism as misuse (M) or anomaly (A) or both (B)
c-denotes the nature of detection as real time (R) or non-real time (N)
d-characterizes the nature of processing as centralized (C) or distributed (D)
e-indicates the data gathering mechanism as centralized (C) or distributed (D)

3.3 Tools Used for Network Traffic Analysis

Capturing and preprocessing high speed network traffic is essential prior to detection of network anomalies. Different tools are used for capturing and analysis of network traffic data. We list a few commonly used tools and their features in Table 3 11. These are commonly used by both the network defender and the attacker at different time points.

Table 3.11: Tools used in different steps in network traffic anomaly detection and their description

Tool Name	*Purpose*	*Characteristics*	*Source*
Wireshark	Packet capture	(i) Free and open-source packet analyzer (ii) Can be used for network troubleshooting, analysis software and communications protocol development, and education (iii) Uses cross-platform GTK+ widget toolkit to implement its user interface and uses pcap to capture packets (iv) Similar to tcpdump, but has a graphical front-end, plus some integrated sorting and filtering options (v) Works in mirrored ports to capture network traffic to analyze for any tampering	http //www wireshark org/
Gulp	Lossless gigabit remote packet capturing	(i) It allows much higher packet capture rate by dropping far fewer packets (ii) It has ability to read directly from the network, but is able to pipe output from legacy applications before writing to disk (iii) If the data rate increases, Gulp realigns its writes to even block boundaries for optimum writing efficiency (iv) 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	http //staff washington edu/corey/gulp/
tcptrace	TCP-based feature extraction	(i) Can take input files produced by several popular packet-capture programs, including tcpdump snoop etherpeek, HP Net Metrix, Wireshark, and WinDump (ii) Produces several types of output containing information on each connection seen such as elapsed time, bytes and segments sent and received, retransmissions, round trip times window advertisements and throughput (iii) Can also produce a number of graphs with packet statistics for further analysis	http //jarok cs ohiou edu/software/tcptrace/
nfdump	netflow data collection	(i) Can collect and process netflow data on the command line (ii) It is limited only by the disk space available for all the netflow data (iii) Can be optimized in speed for efficient filtering The filter rules look like the syntax of tcpdump	http //nfdump sourceforge net/
nfsen	netflow data collection and visualization	(i) nfsen is a graphical Web-based front end for the nfdump netflow tool (ii) It allows display of netflow data as flows, packets and bytes using RRD (Round Robin Database) (iii) Can process the netflow data within a specified time span (iv) Can create history as well as continuous profiles (v) Can set alerts based on various conditions	http //nfsen sourceforge net/
nmap	Scanning port	(i) Network Mapper (nmap) is a free and open source utility for network exploration or security auditing (ii) Uses raw IP packets in novel ways to determine which hosts are available on the network which services (application name and version) those hosts offer, what operating systems are running, type of firewall or packet filter used, and many other characteristics (iii) It is easy flexible, powerful well documented tool for discovering hosts in large network	http //nmap org/
rnmap	Coordinated scanning	(i) Remote Nmap (rnmap) contains both client and server programs (ii) Various clients can connect to one centralized rnmap server and do their port scanning (iii) Server performs user authentication and uses excellent nmap scanner to do actual scanning	http //rnmap sourceforge net/

Continued on next page

Table 3.11 – *Continued from previous page*

Tool Name	*Purpose*	*Characteristics*	*Source*
Targa	Attack simulation	(i) Targa is a free and powerful attack generation tool. (ii) It integrates bonk, jolt, land, nestea. netear, syndrop. teardrop, and winnuke into one multi-platform DoS attack	http.//www10.org/cdrom/papers/409/

3.4 Observations and Summary

The following are some observations that one needs to be mindful of when developing a network anomaly detection method or a system.

- Most existing IDSs for the wired environment work in three ways: flow-level traffic or packet-level feature data analysis, protocol analysis or payload inspection. Each of these categories has its own advantages and limitations. So, a hybridization of these (e.g., protocol level analysis followed by flow level traffic analysis) may give a better performance in terms of known (with high detection rate) as well as unknown attack detection.

- An IDS, to be capable of identifying both known as well as unknown attacks, should exploit both supervised (rule-based or signature-based) as well as unsupervised (clustering or outlier-based) methods at multiple levels for real time performance with low false alarm rates.

- The IDS developer should choose the basic components, method(s), techniques or rule/signature/profile bases to overcome four important limitations: subjective effectiveness, limited scalability, scenario dependent efficiency and restricted security.

- The performance of a better IDS needs to be established both qualitatively and quantitatively.

- A better anomaly classification or identification method enables us to tune it (the corresponding normal profiles, thresholds, etc) depending on the network scenario.

We have examined the state-of-the-art in modern anomaly-based network intrusion detection. The discussion emphasized on two well known criteria to detect anomalous traffic in NIDSs' detection strategy and evaluation. We also presented many detection methods, systems and tools under several categories. Finally, we outlined some recommendations to the future researchers and practitioners who may attempt to develop new detection methods and systems for current network scenarios.

Chapter 4

A Systematic Approach to Generate Real-Life Intrusion Datasets

This chapter is organized in three major sections, viz., Introduction, Existing Datasets, and Dataset Generation. We establish the importance of an intrusion dataset in the development and validation process of a detection mechanism, identify a set of requirements for effective dataset generation, and discuss several attack scenarios. We also describe the motivation and our contribution in Section 4.1. In Section 4.2, we discuss various types of datasets and their characteristics. In the last section of this chapter, we discuss a systematic approach for generation of an unbiased, full feature network intrusion dataset. We also establish the effectiveness of the generated dataset by comparing with several existing datasets.

4.1 Introduction

In network intrusion detection, particularly when using anomaly-based detection, it is difficult to accurately evaluate, compare, and deploy a system that is expected to detect novel attacks due to scarcity of adequate datasets. Before deploying in any real world environment, an anomaly-based network intrusion detection system (ANIDS) must be tested and evaluated using real labelled network traffic traces with a comprehensive set of intrusions or attacks. This is a significant challenge, since not many such datasets are available. Therefore the detection methods and systems are

evaluated only with a few publicly available datasets that lack comprehensiveness and completeness. For example, Cooperative Association for Internet Data Analysis (CAIDA) Distributed Denial of Service (DDoS) 2007, Lawrence Berkeley National Laboratory (LBNL), and ICSI datasets are heavily anonymized without payload information, decreasing research utility. Researchers also frequently use a single NetFlow based intrusion dataset found at [240] with limited number of attacks.

4.1.1 Importance of Datasets

In network traffic anomaly detection, it is always important to test and evaluate detection methods and systems using datasets as network scenarios evolve. We enumerate the following reasons to justify the importance of a dataset.

- *Repeatability of experiments*: Researchers should be able to repeat experiments with the dataset and get similar results, when using the same approach. This is important because the proposed method should cope with the evolving nature of attacks and network scenarios.

- *Validation of new approaches*: New methods and algorithms are being continuously developed to detect network anomalies It is necessary that every new approach be validated.

- *Comparison of different approaches*: State-of-the-art network anomaly detection methods must not only be validated, but also show improvements over older methods in performance in a quantifiable manner For example, the DARPA 1998 dataset [95] is commonly used for performance evaluation of anomaly detection systems [241].

- *Parameters tuning*. To properly obtain the model to classify the normal from malicious traffic, it is necessary to tune model parameters. Network anomaly detection assumes the normality model to identify malicious traffic For example, Cemerlic et al. [242] and Thomas et al. [243] use the attack-free part of the DARPA 1999 dataset for training to estimate parameter values.

- *Dimensionality or the number of features*: An optimal set of features or attributes should be considered to represent normal as well as all possible attack

instances.

4.1.2 Requirements

Although good datasets are necessary for validating and evaluating IDSs, generating such datasets is a time consuming task. A dataset generation approach should meet the following requirements.

- *Real world*: A dataset should be generated by monitoring the daily situation in a realistic way, such as the daily network traffic of an organization.
- *Completeness in labelling*: The labelling of traffic as benign or malicious must be backed by proper evidence for each instance. The aim these days should be to provide labelled datasets at both packet and flow levels for each piece of benign and malicious traffic.
- *Correctness in labelling*: Given a dataset, labelling of each traffic instance must be correct. This means that our knowledge of security events represented by the data has to be certain.
- *Sufficient trace size*: The generated dataset should be unbiased in terms of size in both benign and malicious traffic instances.
- *Concrete feature extraction*: Extraction of an optimal set of concrete features when generating a dataset is important because such features play an important role during validating a detection mechanisms.
- *Diverse attack scenarios*: With the increasing frequency, size, variety, and complexity of attacks, intrusion threats have become more complex including the selection of targeted services and applications. When contemplating attack scenarios for dataset generation, it is important to tilt toward a diverse set of multi-stage attacks that are recent.
- *Ratio between normal and attack traffic*: Most existing datasets have been created based on the following assumptions.
  - Anomalous traffic is statistically different from normal traffic [244].

– The majority of network traffic instances is normal [64]

Unlike most traditional intrusion datasets, DDoS attacks do not follow these assumptions because they change network traffic rate dynamically and employ multi-stage attacks

4.1.3 Motivation and Contributions

By considering the aforementioned requirements, we propose a systematic approach for generating real-life network intrusion dataset at both packet and flow levels in order to analyze, test, and evaluate network intrusion detection methods and systems with a clear focus on anomaly based detectors The following are the major contributions of this chapter

- We present a list of guidelines for real-life intrusion dataset generation
- We discuss systematic generation of both normal and attack traffic
- We extract features from the captured network traffic such as *basic content*-based *time*-based, and *connection*-based features using a distributed feature extraction framework
- We generate three categories of real-life intrusion datasets, viz , (i) TUIDS intrusion dataset, (ii) TUIDS coordinated scan dataset, and (iii) TUIDS DDoS dataset These datasets are available for the research community to download for free

4.2 Existing Datasets

As discussed earlier, datasets play an important role in the testing and validation of network anomaly detection methods or systems A good quality dataset not only allows us to identify the ability of a method or a system to detect anomalous behavior, but also allows us to gave potential effectiveness when deployed in real operating environments Several datasets are publicly available for testing and evaluation of network anomaly detection methods and systems A taxonomy of

network intrusion datasets is shown in Figure 4.1. We briefly discuss each of them below.

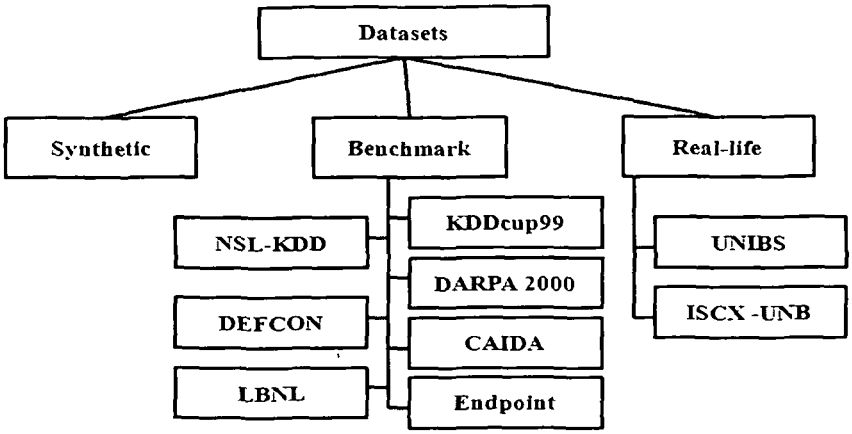


Figure 4.1: A taxonomy of network intrusion datasets

4.2.1 Synthetic Datasets

Synthetic datasets are generated to meet specific needs or certain conditions or tests that real data satisfy. Such datasets are useful when designing any prototype system for theoretical analysis so that the design can be refined. As stated previously, a synthetic dataset can be used to test and create many different types of test scenarios. This enables designers to build realistic behavior profiles for normal users and attackers based on the dataset to test a proposed system. This provides initial validation of a specific method or a system; if the results prove to be satisfactory, the developers then continue to evaluate a method or a system in a specific domain.

4.2.2 Benchmark Datasets

We discuss seven publicly available benchmark datasets generated using simulated environments in large networks, executing different attack scenarios.

KDDcup99 Dataset

Since 1999, the KDDcup99 dataset [52] has been the most widely used dataset for evaluation of network based anomaly detection methods and systems. This dataset was prepared by Stolfo et al. [245] and is built upon the data captured

in the DARPA98 IDS evaluation program. The KDD training dataset consists of approximately 4,900,000 single connection vectors, each of which contains 41 features and is labeled as either normal or attack of a specific attack type. The test dataset contains about 300,000 samples with a total 24 training attack types, with an additional 14 attack types in the test dataset only [43]. The represented attacks are mainly four types: denial of service, remote-to-local, user-to-root, and surveillance or probing.

- *Denial of Service (DoS)*: An attacker attempts to prevent valid users from using a service provided by a system. Examples include SYN flood, smurf and teardrop attacks.
- *Remote to Local (r2l)*: Attackers try to gain entrance to a victim machine without having an account on it. An example is the password guessing attack.
- *User to Root (u2r)*: Attackers have access to a local victim machine and attempt to gain privilege of a superuser. Examples include buffer overflow attacks.
- *Probe*: Attackers attempt to acquire information about the target host. Some examples of probe attacks are port-scans, and ping-sweep attacks.

Background traffic was simulated and the attacks were all known. The training set, consisting of seven weeks of labeled data, is available to the developers of intrusion detection systems. The testing set also consists of simulated background traffic and known attacks, including some attacks that are not present in the training set. The distribution of normal and attack traffic for this dataset is reported in Table 4.1. We also identify the services associated with each category of attacks [246,247] and summarize them in Table 4.2.

NSL-KDD Dataset

Analysis of the KDD dataset showed that there were two important issues with the dataset, which highly affect the performance of evaluated systems resulting in poor evaluation of anomaly detection methods [248]. To address these issues, a new

Table 4.1: Distribution of normal and attack traffic instances in KDDCup99 dataset

Dataset	*DoS*		*Probe*		*u2r*		*r2l*		Normal
	Total instances	Attacks	Total instances	Attacks	Total instances	Attacks	Total instances	Attacks	
10% KDD	391458	smurf, neptune, back, teardrop, pod, land	4107	satan, ipsweep, portsweep, nmap	52	buffer_overflow, rootkit, loadmodule, perl	1126	warezclient, guess_passwd, warezmaster, imap, ftp_write, multihop phf, spy	97277
Corrected KDD	229853		4107		52		1126		97277
Whole KDD	229853		4107		52		1126		97277

Table 4.2: List of attacks and corresponding services in KDDcup99 dataset

Dataset	*DoS*		*Probe*		*u2r*		*r2l*	
	Attack name	Service(s)	Attack name	Service(s)	Attack name	Service(s)	Attack name	Service(s)
KDD99	apache2	http	ipsweep	icmp	eject	Any user session	dictionary	telnet, rlogin, pop imap, ftp
	back	http	mscan	many	ffbconfig	Any user session	ftp-write	ftp
	land	N/A	nmap	many	fdformat	Any user session	guest	telnet, rlogin
	mailbomb	smtp	saint	many	loadmodule	Any user session	imap	imap
	SYN flood	Any TCP	satan	many	perl	Any user session	named	dns
	ping of death	icmp			ps	Any user session	named	dns
	process table	Any TCP			Xterm	Any user session	sendmail	smtp
	smurf	icmp	-	-			xlock	X
	syslogd	syslog					xsnoop	X
	teardrop	N/A			-	-		
	udpstorm	echo/chargen					-	-

dataset known as NSL-KDD [249], consisting of selected records of the complete KDD dataset was introduced. This dataset is publicly available for researchers[1] and has the following advantages over the original KDD dataset.

- It does not include redundant records in the training set, so classifiers will not be biased towards more frequent records.

- There are no duplicate records in the test set. Therefore, the performance of learners is not biased by the methods which have better detection rates on frequent records.

- The number of selected records from each difficulty level is inversely proportional to the percentage of records in the original KDD dataset. As a result, the classification rates of various machine learning methods vary in a wider range, which makes it more efficient to have an accurate evaluation of various learning techniques.

- The number of records in the training and testing sets is reasonable, which makes it affordable to run experiments on the complete set without the need to randomly select a small portion. Consequently, evaluation results of different research groups are consistent and comparable

The NSL-KDD dataset consists of two parts: (i) KDDTrain$^+$ and (ii) KDDTest$^+$. The distribution of attack and normal instances in the NSL-KDD dataset is shown in Table 4.3.

Table 4.3: Distribution of normal and attack traffic instances in NSL-KDD dataset

Dataset	*DoS*	*u2r*	*r2l*	*Probe*	*Normal*	*Total*
KDDTrain$^+$	45927	52	995	11656	67343	125973
KDDTest$^+$	7458	67	2887	2422	9710	22544

[1] http://www.iscx.ca/NSL-KDD/

DARPA 2000 Dataset

A DARPA[1] evaluation project [250] targeted the detection of complex attacks that contain multiple steps. Two attack scenarios were simulated in the DARPA 2000 evaluation contest, namely Lincoln Laboratory scenario DDoS (LLDOS) 1.0 and LLDOS 2.0. To achieve variations, these two attack scenarios were carried out over several network and audit scenarios. These sessions were grouped into four attack phases: (a) probing, (b) breaking into the system by exploiting vulnerability, (c) installing DDoS software for the compromised system, and (d) launching DDoS attack against another target. LLDOS 2.0 is different from LLDOS 1.0 in that attacks are more stealthy and thus harder to detect. Since this dataset contains multistage attack scenarios, it is also commonly used for evaluation of alert correlation techniques.

DEFCON Dataset

The DEFCON[2] dataset is another commonly used dataset for evaluation of IDSs [251]. It contains network traffic captured during a hacker competition called Capture The Flag (CTF), in which competing teams are divided into two groups: *attackers* and *defenders*. The traffic produced during CTF is very different from real world network traffic since it contains only intrusive traffic without any normal background traffic. Due to this limitation, DEFCON dataset has been found useful only in evaluating alert correlation techniques

CAIDA Dataset

CAIDA[3] collects many different types of data and makes them available to the research community. CAIDA datasets [252] are very specific to particular events or attacks. Most of its longer traces are anonymized backbone traces without their payload. The CAIDA DDoS 2007 attack dataset contains one hour of anonymized traffic traces from DDoS attacks on August 4, 2007, which attempted to consume a large amount of network resources when connecting to Internet servers. The traffic

[1] http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/data/index.html
[2] http://cctf.shmoo.com/data/
[3] http://www.caida.org/home/

traces contain only attack traffic to the victim and responses from the victim with 5 minutes split form. All traffic traces are in pcap (tcpdump) format. The creators removed non-attack traffic as much as possible when creating the CAIDA DDoS 2007 dataset.

LBNL Dataset

LBNL's internal enterprise traffic traces are full header network traces [253] without payload. This dataset suffers from heavy anonymization to the extent that scanning traffic was extracted and separately anonymized to remove any information which could identify individual IPs. The background and attack traffic in the LBNL dataset are described below.

- *LBNL background traffic:* This dataset can be obtained from the Lawrence Berkeley National Laboratory (LBNL) in the US. Traffic in this dataset is comprised of packet level incoming, outgoing, and internally routed traffic streams at the LBNL edge routers. Traffic was anonymized using the *tcpmkpub* tool [254]. The main applications observed in the internal and external traffic are Web, email, and name services. Other applications like Windows services, network file services, and backup were used by internal hosts. The details of each service and information on each packet and other relevant description are given in [255]. The background network traffic statistics of LBNL dataset are given in Table 4.4.

- *LBNL attack traffic:* This dataset identifies attack traffic by isolating scans in aggregate traffic traces. Scans are identified by flagging those hosts which unsuccessfully probe more than 20 hosts, out of which 16 hosts are probed in ascending or descending IP order [254]. Malicious traffic mostly consists of failed incoming TCP SYN requests, i.e., TCP port scans targeted towards LBNL hosts. However, there are also some outgoing TCP scans in the dataset. Most UDP traffic observed in the data (incoming and outgoing) is comprised of successful connections, i.e., host replies for the received UDP flows. Clearly, the attack rate is significantly lower than the background traffic rate. Details of the attack traffic in this dataset are shown in Table 4.4. Complexity and

Table 4.4: Background and attack traffic information for the LBNL datasets

Date	*Duration (mins)*	*LBNL hosts*	*Remote hosts*	*Background traffic rate (packet/sec)*	*Attack traffic rate (packet/sec)*
10/04/2004	10 min	4 767	4.342	8 47	0 41
12/15/2004	60 min	5,761	10,478	3.5	0 061
12/16/2004	60 min	5,210	7,138	243 83	72

privacy were two main reservations of the participants of the endpoint data collection study. To address these reservations, the dataset creators developed a custom multi-threaded MS Windows tool using the *Winpcap* API [256] for data collection. To reduce packet logging complexity at the endpoints, they only logged very elementary session-level information (bidirectional communication between two IP addresses on different ports) for the TCP and UDP packets To ensure user privacy, an anonymization policy was used to anonymize all traffic instances.

Endpoint Dataset

The background and attack traffic for the endpoint datasets are explained below.

- *Endpoint background traffic:* In the endpoint context, we see in Table 4.5 that home computers generate significantly higher traffic volumes than office and university computers because: (*i*) they are generally shared between multiple users, and (*ii*) they run peer-to-peer and multimedia applications. The large traffic volumes of home computers are also evident from their high mean number of sessions per second To generate attack traffic, the developers infected Virtual Machines (VMs) on the endpoints with different malware, viz., Zotob.G, Forbot-FU, Sdbot-AFR, Dloader-NY, So-Big.E@mm, MyDoom.A@mm, Blaster, Rbot-AQJ, and RBOT.CCC. Details of the malware can be found in [257]. Characteristics of the attack traffic in this dataset are given in Table 4.6. These malwares have diverse scanning rates and attack ports or applications.

- *Endpoint attack traffic:* The attack traffic logged at the endpoints is mostly comprised of outgoing port scans. Note that this is the opposite of the LBNL

Table 4.5: Background traffic information for four endpoints with high and low rates

Endpoint ID	*Endpoint type*	*Duration (months)*	*Total sessions*	*Mean session rate (/sec)*
3	Home	3	3.73,009	1.92
4	home	2	4 44,345	5 28
6	University	9	60,979	0.19
10	University	13	1 52,048	0 21

Table 4.6: Endpoint attack traffic for two high and two low-rate worms

Malware	*Release Date*	*Avg Scan rate (/sec)*	*Port (s) Used*
Dloader-NY	Jul 2005	46.84 sps	TCP 1,35,139
Forbot-FU	Sept 2005	32 53 sps	TCP 445
Rbot-AQJ	Oct 2005	0 68 sps	TCP 1,39.769
MyDoom-A	Jan 2006	0.14 sps	TCP 3127-3198

dataset, in which most attack traffic is inbound. Moreover, the attack traffic rates at the endpoints are generally much higher than the background traffic rates of the LBNL datasets This diversity in attack direction and rates provides a sound basis for performance comparison among scan detectors. For each malware, attack traffic of 15 minute duration was inserted in the background traffic for each endpoint at a random time instance. This operation was repeated to insert 100 non-overlapping attacks of each worm inside each endpoint's background traffic

4.2.3 Real-life Datasets

We discuss three real-life datasets created by collecting network traffic on several consecutive days during a week or a month. The details include both normal as well as attack traffic in appropriate proportions in the authors' respective campus networks (i.e., testbed).

UNIBS Dataset

The UNIBS packet traces [258] were collected on the edge router of the campus network of the University of Brescia in Italy, on three consecutive working days. The dataset includes traffic captured or collected and stored using 20 workstations, each running the GT (Ground Truth) client daemon. The dataset creators collected

the traffic by running tcpdump on the faculty router, which was a dual Xeon Linux box that connected the local network to the Internet through a dedicated 100Mb/s uplink. They captured and stored the traces on a dedicated disk of a workstation connected to the router through a dedicated ATA controller.

ISCX-UNB Dataset

Real packet traces [259] were analyzed to create profiles for agents that generate real traffic for HTTP, SMTP, SSH, IMAP, POP3 and FTP protocols. Various multistage attack scenarios were explored to generate malicious traffic.

KU Dataset

The Kyoto University dataset[1] is a collection of network traffic data obtained from honeypots. The raw dataset obtained from the honeypot system consisted of 24 statistical features, out of which 14 significant features were extracted [260]. The dataset developers extracted 10 additional features that could be used to investigate network events inside the university more effectively. However, they used 14 conventional features only during training and testing.

4.2.4 Discussion

The datasets described above are valuable assets for the intrusion detection community. However, the benchmark datasets suffer from the fact that they are not good representatives of real world traffic. For example, the DARPA dataset has been questioned about the realism of the background traffic [261, 262] because it is synthetically generated. In addition to the difficulty of simulating real-life network traffic, there are additional challenges in IDS evaluation [263]. These include difficulties in collecting attack scripts and victim software, differing requirements for testing signature based vs. anomaly based IDSs, and host-based vs. network based IDSs.

[1] http://www.takakura.com/kyoto_data

4.3 Real-Life Datasets Generation

As noted above, the generation of an unbiased real-life intrusion dataset incorporating a large number of real world attacks is important to evaluate network anomaly detection methods and systems In this chapter, we describe the generation of three real-life network intrusion datasets[1] including (a) a TUIDS (Tezpur University Intrusion Detection System) intrusion dataset, (b) a TUIDS coordinated scan dataset, and (c) a TUIDS DDoS dataset at both packet and flow levels [264] The resulting details and supporting infrastructure is discussed in the following subsections

4.3.1 Testbed Network Architecture

The TUIDS testbed network consists of 250 hosts 15 L2 switches 8 L3 switches, 3 wireless controllers, and 4 routers that compose 5 different networks inside the Tezpur University campus The architecture of the TUIDS testbed is given in Figure 4 2 The hosts are divided into several VLANs each VLAN belonging to an L3 switch or an L2 switch inside the network All servers are installed inside a DMZ to provide an additional layer of protection in the security system of an organization

4.3.2 Network Traffic Generation

To generate real-life normal and attack traffic we configured several hosts, workstations and servers in the TUIDS testbed network The network consists of 6 interconnected Ubuntu 10 10 workstations On each workstation, we have installed several severs including a network file server (Samba) a mail sever (Dovecot), a telnet server, an FTP server a Web server, and an SQL sever with PHP compatibility We also installed and configured 4 Windows Servers 2003 to exploit a diverse set of known vulnerabilities against the testbed environment Servers and their services running in our testbed are summarized in Table 4 7

The normal network traffic is generated based on the day-to-day activities of users and especially generated traffic from configured servers It is important to generate different types of normal traffic So, we capture traffic from students,

[1] http //agnigarh tezu ernet in/~dkb/resource html

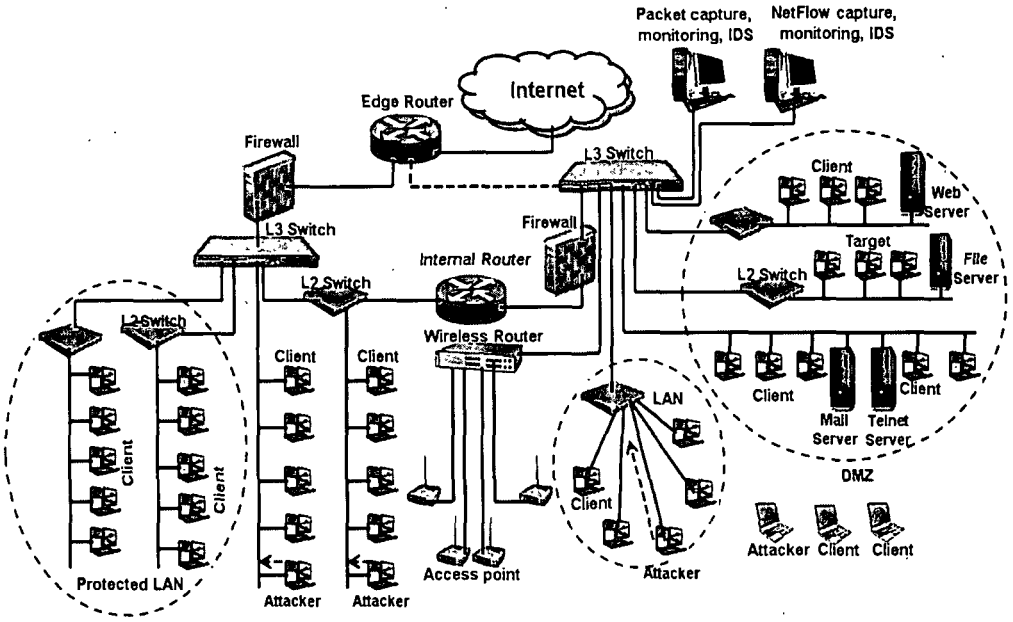


Figure 4.2: Testbed network architecture

Table 4.7: Servers and their services running on the testbed network

Server	*Operating system*	*Services*	*Provider*
Main Server	Ubuntu 10.10	Web, eMail	Apache 2.4.3, Dovecot 2.1.14
Network File Server	Ubuntu 10.10	Samba	Samba 4.0.2
Telnet Server	Ubuntu 10.10	Telnet	telnet-0.17-36bulid1
FTP Server	Ubuntu 10.10	ftp	vsFTPd 2.3.0
Windows Server	Windows Server 2003	Web	IIS v7.5
MySQL Server	Ubuntu 10.10	database	MySQL 5.5.30

faculty members, system administrators, and office staff on different days within the University. The attack traffic is generated by launching attacks within the testbed network in three different subsets, viz., a TUIDS intrusion dataset, a coordinated scan dataset, and a DDoS dataset. The attacks launched in the generation of these real-life datasets are summarized in Table 4.8.

As seen in the table above, 22 distinct attack types (1-22 in Table 4.8) were used to generate the attack traffic for the TUIDS intrusion dataset; six attacks (17-22 in Table 4.8) were used to generate the attack traffic for the coordinated scan dataset and finally six attacks (23-28 in Table 4.8) were used to generate the attack traffic for a DDoS dataset with combination of TCP, UDP, and ICMP protocols.

Table 4.8: List of real-life attacks and their generation tools

Attack name	*Generation tool*	*Attack name*	*Generation tool*
1 bonk	targa2 c	15 linux-icmp	linux-icmp c
2 jolt	targa2 c	16 syn-flood	synflood c
3 land	targa2 c	17 window-scan	nmap/rnmap
4 saihyousen	targa2 c	18 syn-scan	nmap/rnmap
5 teardrop	targa2 c	19 xmasstree-scan	nmap/rnmap
6 newtear	targa2 c	20 fin-scan	nmap/rnmap
7 1234	targa2 c	21 null-scan	nmap/rnmap
8 winnuke	targa2 c	22 udp-scan	nmap/rnmap
9 oshare	targa2 c	23 syn-flood(DDoS)	LOIC
10 nestea	targa2 c	24 rst-flood(DDoS)	Trinity v3
11 syndrop	targa2 c	25 udp-flood(DDoS)	LOIC
12 smurf	smurf4 c	26 ping-flood(DDoS)	DDoS ping v2 0
13 opentear	opentear c	27 fraggle udp-flood(DDoS)	Trinoo
14 fraggle	fraggle c	28 smurf icmp-flood(DDoS)	TFN2K

4.3.3 Attack Scenarios

The attack scenarios start with information gathering techniques collecting target network IP ranges, identities of name servers, mail servers, and user e-mail accounts, etc This is achieved by querying the DNS for resource records using network administrative tools like nslookup, and dig We consider six attack scenarios when collecting real-life network traffic for dataset generation

Scenario 1: Denial of Service using targa

This attack scenario is designed towards performing attacks on a target using the targa[1] tool until it is successful Targa is a very powerful tool to quickly damage a particular network belonging to an organization We ran targa by specifying different parameter values such as IP ranges, attacks to run, and number of times to repeat the attack

Scenario 2: Probing using nmap

In this scenario, we attempt to acquire information about the target host and then launch the attack by exploiting the vulnerabilities found using the nmap[2] tool

[1] http //packetstormsecurity com/
[2] http //nmap org/

Examples of attacks that can be launched by this method are syn-scan and ping-sweep.

Scenario 3: Coordinated scan using rnmap

This scenario starts with a goal to perform coordinated port scans to single and multiple targets. Tasks are distributed among multiple hosts for individual actions which may be synchronized. We use the rnmap[1] tool to launch coordinated scans in our testbed network during the collection of traffic.

Scenario 4: User to root using brute force ssh

These attacks are very common against networks as they tend to break into accounts with weak username and password combinations. This attack has been designed with the goal of acquiring an SSH account by running a dictionary brute force attack against our central server. We use the brutessh[2] tool and a customized dictionary list. The dictionary consists of over 6100 alphanumeric entries of varying length. We executed the attack for 60 minutes, during which superuser credentials were returned from the server. This ID and password combination was used to download other users' credentials immediately.

Scenario 5: Distributed Denial of Service using agent-handler network

This scenario mainly attempts to exploit an agent handler network to launch the DDoS attack in the TUIDS testbed network. The agent-handler network consists of clients, handlers, and agents. The handlers are software packages that are used by the attacker to communicate indirectly with the agents. The agent software exists in compromised systems that will eventually carry out the attack on the victim system. The attacker may communicate with any number of handlers, thus making sure that the agents are up and running. We use Trinity v3, TFN2K, Trinoo, and DDoS ping 2.0 to launch the attacks in our testbed.

[1] http://rnmap.sourceforge.net/
[2] http://www.securitytube-tools.net/

Scenario 6: **Distributed Denial of Service using IRC botnet**

Botnets are an emerging threat to all organizations because they can compromise a network and steal important information and distribute malware. Botnets combine individual malicious behaviors into a single platform by simplifying the actions needed to be performed by users to initiate sophisticated attacks against computers or networks around the world. These behaviors include coordinated scanning, distributed denial of service (DDoS) activities. direct attacks, indirect attacks, and other deceitful activities taking place across the Internet.

The main goal of this scenario is to perform distributed attacks using infected hosts on the testbed. Internet relay chat (IRC) bot network allow users to create public, private and secret channels. For this, we use a LOIC[1], an IRC-based DDoS attack generation tool. The IRC systems have several other significant advantages for launching DDoS attacks. Among the three important benefits are (i) they afford a high degree of anonymity, (ii) they are difficult to detect, and (iii) they provide a strong, guaranteed delivery system. Furthermore, the attacker no longer needs to maintain a list of agents, since he can simply log on to the IRC server and see a list of all available agents The IRC channels receive communications from the agent software regarding the status of the agents (i.e., up or down) and participate in notifying the attackers regarding the status of the agents.

4.3.4 Capturing Traffic

The key tasks in network traffic monitoring are lossless packet capturing and precise time stamping. Therefore, software or hardware is required with a guarantee that all traffic is captured and stored. With the goal of preparing both packet and flow level datasets, we capture both packet and NetFlow traffic from different locations in the TUIDS testbed. The capturing period started at 08:00:05am on Monday February 21, 2011 and continuously ran for an exact duration of seven days, ending at 08:00:05am on Sunday February 27th. Attacks were executed during this period for the TUIDS Intrusion and the Coordinated Scan datasets. DDoS traffic was also collected for the same amount of time but during October, 2012 with several

[1]http //sourceforge.net/projects/loic/

variations of real-life DDoS attacks. Figure 4.3 illustrates the protocol composition and the average throughput seen during the last hour of data capture for the TUIDS intrusion dataset in our lab

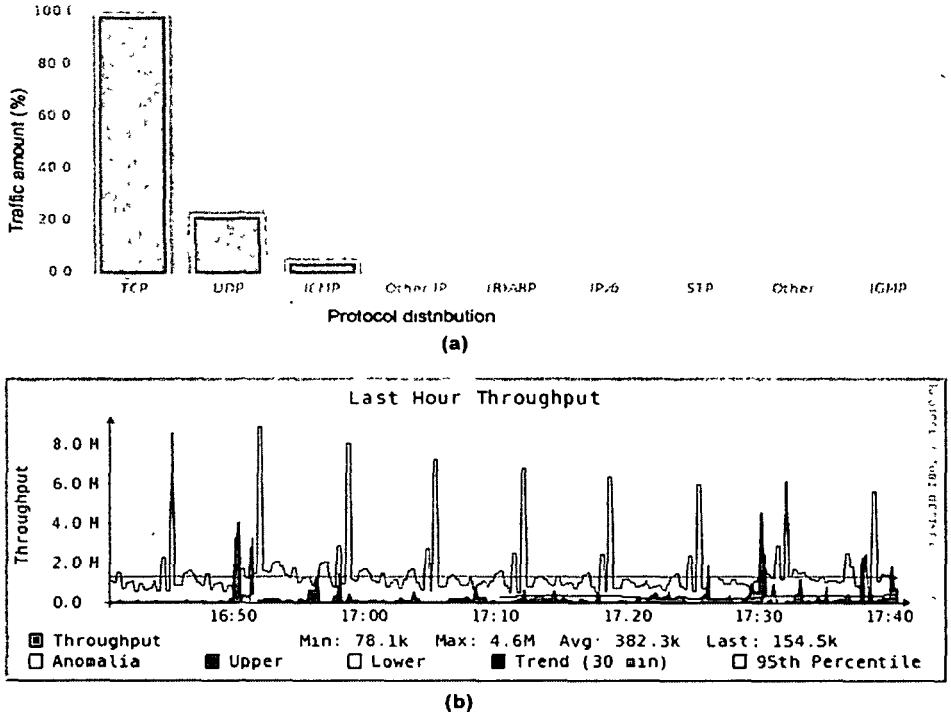


Figure 4.3: (a) Composition of protocols and (b) Average throughput during last hour of data capture for the TUIDS intrusion dataset seen in our lab's traffic

We use a tool known as Lossless Gigabit Remote Packet Capture with Linux (Gulp[1]) for capturing packet level traffic in a mirror port as shown in the TUIDS testbed architecture. Gulp reads packets directly from the network card and writes to the disk at a high rate of packet capture without dropping packets. For low rate packets, Gulp flushes the ring buffer if it has not written anything in the last second. Gulp writes into even block boundaries for excellent writing performance when the data rate increases. It stops filling the ring buffer after receiving an interrupt but it would write into the disk whatever remains in the ring buffer.

In the last few years, NetFlow has become the most popular approach for IP network monitoring, since it helps cope with the scalability issues introduced by increasing network speeds. Now major vendors offer flow-enabled devices. An example is a Cisco router with NetFlow. A NetFlow is a stream of packets that

[1]http://staff.washington.edu/corey/gulp/

arrives on a source interface with the key values shown in Figure 4 4 A key is an identified value for a field within the packet Cisco routers have NetFlow features that can be enabled to generate NetFlow records The principle of NetFlow is as follows When the router receives a packet, its NetFlow module scans the source IP address, the destination IP address, the source port number, the destination port number the protocol type, the type of service (ToS) bit in IP header, and the input or output interface number on the router of the IP packet, to judge whether it belongs to a NetFlow record that already exists in the cache If so, it updates the NetFlow record, otherwise, a new NetFlow record is generated in the cache The expired NetFlow records in the cache are exported periodically to a destination IP address using a UDP port

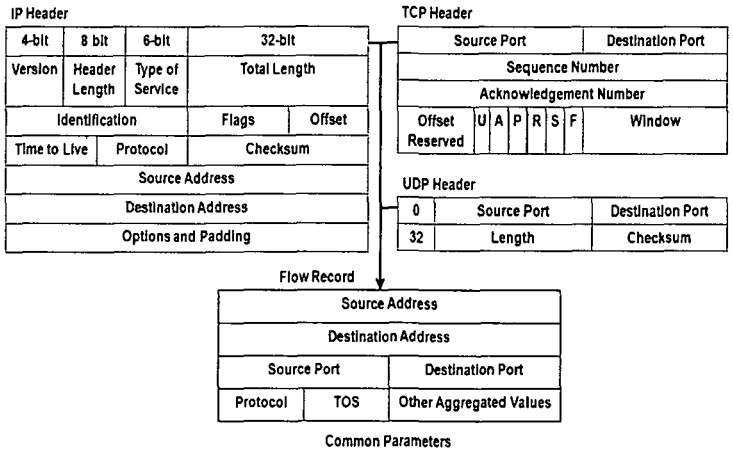


Figure 4.4: Common NetFlow parameters

For capturing NetFlow traffic, we need a NetFlow collector that can listen to a specific UDP port to collect traffic The NetFlow collector captures exported traffic from multiple routers and periodically stores it in summarized or aggregated format into a round robin database (RRD) The following tools are used to capture and visualize the NetFlow traffic

(a) *NFDUMP* This tool captures and displays NetFlow traffic All versions of nfdump support NetFlow v5 v7, and v9 nfcapd is a NetFlow capture daemon that reads the NetFlow data from the routers and stores the data into files periodically It automatically rotates files every n minutes (by default it is 5 minutes) We need one nfcapd process for each NetFlow stream Nfdump reads the NetFlow

data from the files stored by nfcapd The syntax is similar to that of tcpdump Nfdump displays NetFlow data and can create top N statistics for flows based on the parameters selected The main goal is to analyze NetFlow data from the past as well as to track interesting traffic patterns continuously from high speed networks The amount of time from the past is limited only by the disk space available for all NetFlow data

Nfdump has four fixed output formats *raw line long* and *extended* In addition the user may specify any desired output format by customizing it The default format is line, unless specified The raw format displays each record in multiple lines, and prints any available information in the traffic record

(b) *NFSEN* nfsen is a graphical Web based front end tool for visualization of NetFlow traffic nfsen facilitates the visualization of several traffic statistics, e g , flow-wise statistics for various features, navigation through the NetFlow traffic processes within a time span, and continuous profiles It can also add own plugins to process NetFlow traffic in a customized manner at a regular time interval

Normal traffic is captured by restricting it to the internal networks, where 80% of the hosts are connected to the router, including wireless networks We assume that normal traffic follows the normal probability distribution Attack traffic is captured as we launch various attacks in the testbed for a week For DDoS attacks we used packet-craft[1] to generate customized packets Figure 4 5 and Figure 4 6 show the number of flows per second and also the protocol-wise distribution of flows during the capturing period, respectively

4.3.5 Feature Extraction

We use wireshark and Java routines for filtering unwanted packets (such as packets with routing protocols, and packets with application layer protocols) as well as irrelevant information from the captured packets Finally, we retrieve all relevant information from each packet using Java routines and store it in comma-separated form in a text file The details of parameters identified for packet level data are shown in Table 4 9

[1]http //www packet-craft net/

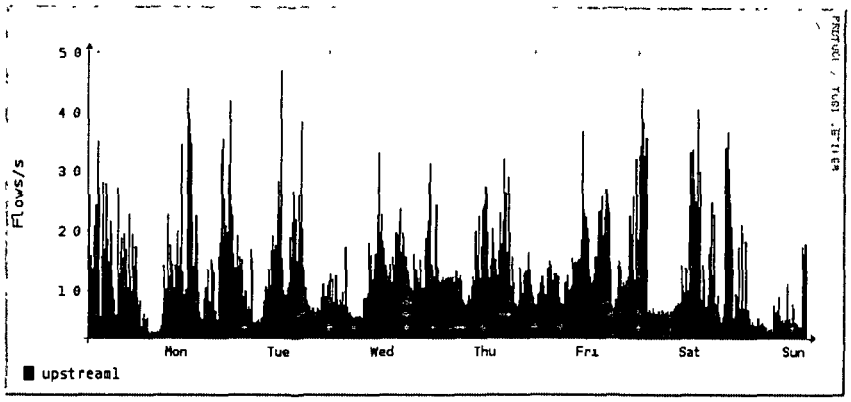


Figure 4.5: Number of flows per second in TUIDS intrusion datasets during the capture period

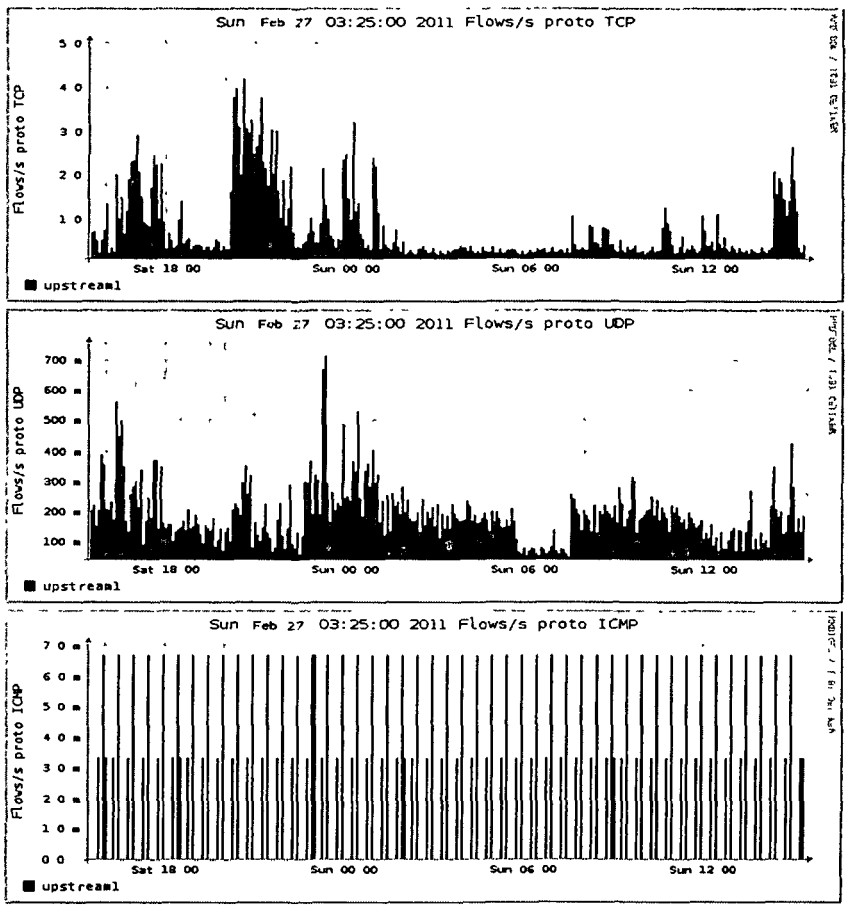


Figure 4.6: Protocol-wise distribution of flow per second in TUIDS intrusion dataset during the capture period

We developed several C routines and used them for filtering NetFlow data and for extracting features from the captured data. A detailed list of parameters identified for flow level data is given in Table 4.10.

Table 4.9: Parameters identified for packet level data

Sl. No.	*Parameter name*	*Description*
1	Time	Time since occurrence of first frame
2	Frame-no	Frame number
3	Frame-len	Length of a frame
4	Capture-len	Capture length
5	TTL	Time to live
6	Protocol	Protocols (such as, TCP, UDP, ICMP etc.)
7	Src-ip	Source IP address
8	Dst-ip	Destination IP address
9	Src-port	Source port
10	Dst-port	Destination port
11	Len	Data length
12	Seq-no	Sequence number
13	Header-len	Header length
14	CWR	Congestion window record
15	ECN	Explicit congestion notification
16	URG	Urgent TCP flag
17	ACK	Acknowledgement flag
18	PSH	Push flag
19	RST	Reset flag
20	SYN	TCP syn flag
21	FIN	TCP fin flag
22	Win Size	Window Size
23	MSS	Maximum segment size

Table 4.10: Parameters identified for flow level data

Sl. No.	*Parameter name*	*Description*
1	flow-start	Starting of flow
2	Duration	Total life time of a flow
3	Proto	Protocol, i.e., TCP, UDP, ICMP, etc.
3	Src-ip	Source IP address
4	Src-port	Source port
5	Dest-ip	Destination IP address
6	Dest-port	Destination port
7	Flags	TCP flags
8	ToS	Type of Service
9	Packets	Packets per flow
10	Bytes	Bytes per flow
11	Pps	Packet per second
12	Bps	Bit per second
13	Bpp	Byte per packet

We capture, preprocess, and extract various features in both packet and flow level network traffic. We introduce a framework for fast distributed feature extraction from raw network traffic, correlation computation and data labelling, as shown in Figure 4.7. We extract four types of features: *basic*, *content*-based, *time*-based and *connection*-based, from the raw network traffic. We use $T = 5$ seconds as the time window for extraction of both time based and connection based traffic

features. S_1 and S_2 are servers used for preprocessing, attack labelling, and profile generation. WS_1 and WS_2 are high-end workstations used for basic feature extraction and merging packet and NetFlow traffic $N_1, N_2, \cdots N_6$ are independent nodes used for protocol specific feature extraction. The lists of extracted features at both packet and flow levels for the intrusion datasets are presented in Table 4.11 and Table 4.12, respectively. The list of features available in the KDDcup99 intrusion dataset is also shown in Table 4.13.

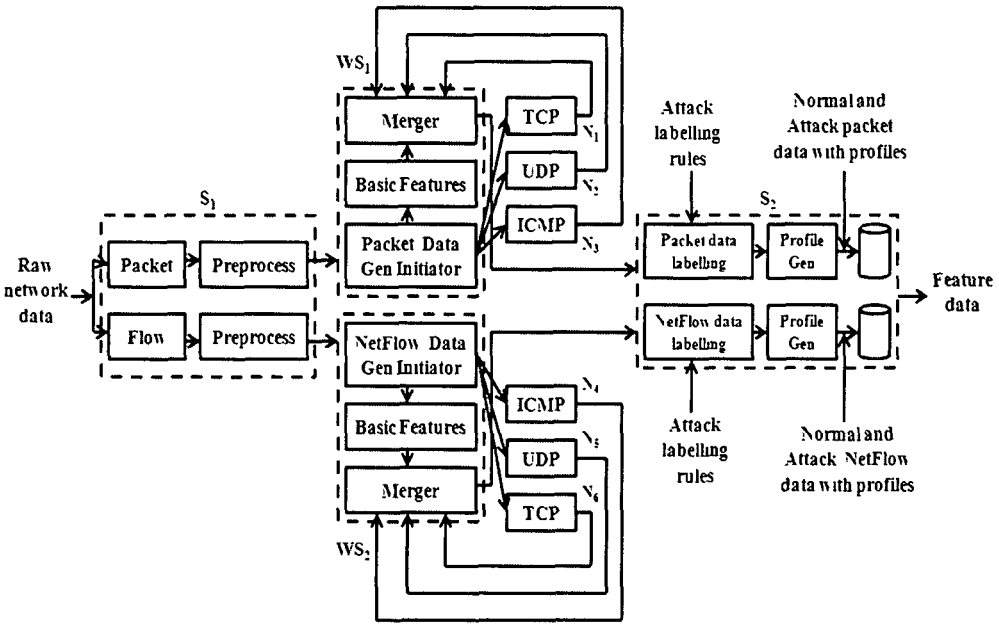


Figure 4.7: Fast distributed feature extraction, correlation, and labelling framework

Table 4.11: List of packet level features in TUIDS Intrusion Dataset

Label/feature name	*Type*	*Description*
Basic features		
1 Duration	C	Length (number of seconds) of the connection
2 Protocol-type	D	Type of protocol, e g , tcp, udp, etc
3 Src-ip	C	Source host IP address
4 Dest-ip	C	Destination IP address
5 Src-port	C	Source host port number
6 Dest-port	C	Destination host port number
7 Service	D	Network service at the destination e g , http, telnet etc
8 num-bytes-src-dst	C	The number of data bytes flowing from source to destination
9 num-bytes-dst-src	C	The number of data bytes flowing from destination to source
10 Fr-no	C	Frame number
11 Fr-len	C	Frame length
12 Cap-len	C	Captured frame length
13 Head-len	C	Header length of the packet
14 Frag-off	D	Fragment offset '1 for the second packet overwrite everything, '0' otherwise

Continued on next page

Table 4.11 – *Continued from previous page*

Label/feature name	*Type*	*Description*
15 TTL	C	Time to live '0 discards the packet
16 Seq-no	C	Sequence number of the packet
17 CWR	D	Congestion window record
18 ECN	D	Explicit congestion notification
19 URG	D	Urgent TCP flag
20 ACK	D	Acknowledgement flag value
21 PSH	D	Push TCP flag
22 RST	D	Reset TCP 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
Content-based features		
26 Mss-src-dest-requested	C	Maximum segment size from source to destination requested
27 Mss-dest-src-requested	C	Maximum segment size from destination to source requested
28 Ttt-len-src-dst	C	Time to live length from source to destination
29 Ttt-len-dst-src	C	Time to live length from destination to source
30 Conn-status	C	Status of the connection (e g , '1' for complete, 0' for reset)
Time-based features		
31 count-fr-dest	C	Number of frames received by unique destinations in the last T seconds from the same source
32 count-fr-src	C	Number of frames received from unique sources in the last T seconds from the same destination
33 count-serv-src	C	Number of frames from the source to the same destination port in the last T seconds
34 count-serv-dest	C	Number of frames from destination to the same source port in the last T seconds
35 num-pushed-src-dst	C	The number of pushed packets flowing from source to destination
36 num-pushed-dst-src	C	The number of pushed packets flowing from destination to source
37 num-SYN-FIN-src-dst	C	The number of SYN/FIN packets flowing from source to destination
38 num-SYN-FIN-dst-src	C	The number of SYN/FIN packets flowing from destination to source
39 num-FIN-src-dst	C	The number of FIN packets flowing from source to destination
40 num-FIN-dst-src	C	The number of FIN packets flowing from destination to source
Connection-based features		
41 count-dest-conn	C	Number of frames to unique destinations in the last N packets from the same source
42 count-src-conn	C	Number of frames from unique sources in the last N packets to the same destination
43 count-serv-srcconn	C	Number of frames from the source to the same destination port in the last N packets
44 count-serv-destconn	C	Number of frames from the destination to the same source port in the last N packets
45 num-packets-src-dst	C	The number of packets flowing from source to destination
46 num-packets-dst-src	C	The number of packets flowing from destination to source
47 num-acks-src-dst	C	The number of acknowledgement packets flowing from source to destination
48 num-acks-dst-src	C	The number of acknowledgement packets flowing from destination to source
49 num-retransmit-src-dst	C	The number of retransmitted packets flowing from source to destination
50 num-retransmit-dst-src	C	The number of retransmitted packets flowing from destination to source
C-Continuous, D-Discrete		

Table 4.12: List of flow level features in TUIDS Intrusion Dataset

Label/feature name	*Type*	*Description*
Basic features		
1 Duration	C	Length (number of seconds) of the flow
2 Protocol-type	D	Type of protocol, e g TCP, UDP, ICMP
3 Src-ip	C	Source host IP address
4 Dest-ip	C	Destination IP address
5 Src-port	C	Source host port number
6 Dest-port	C	Destination host port number
7 ToS	D	Type of service
8 URG	D	TCP urgent flag
9 ACK	D	TCP acknowledgement flag
10 PSH	D	TCP push flag
11 RST	D	TCP reset flag
12 SYN	D	TCP SYN flag
13 FIN	D	TCP FIN flag
14 Src-bytes	C	Number of data bytes transfered from source to destination
15 Dest-bytes	C	Number of data bytes transfered from destination to source
16 Land	D	1 if connection is from/to the same host/port, 0 otherwise
Time-based features		
17 count-dest	C	Number of flows to unique destination IPs in the last T seconds from the same source
18 count-src	C	Number of flows from unique source IPs in the last T seconds to the same destination
19 count-serv-src	C	Number of flows from the source to the same destination port in the last T seconds
20 count-serv-dest	C	Number of flows from the destination to the same source port in the last T seconds
Connection-based features		
21 count-dest-conn	C	Number of flows to unique destination IPs in the last N flows from the same source
22 count-src-conn	C	Number of flows from unique source IPs in the last N flows to the same destination
24 count-serv-srcconn	C	Number of flows from the source IP to the same destination port in the last N flows
25 count-serv-destconn	C	Number of flows to the destination IP to the same source port in the last N flows
C-Continuous, D-Discrete		

4.3.6 Data Processing and Labelling

As mentioned in the previous section, both packet and flow level traffic features are extracted separately within a time interval when features are extracted. So, it is important to correlate each feature (i.e., basic, content-based, time-based, and connection-based) to a time interval. Once correlation is performed for both packet and flow level traffic, labelling of each feature data as normal or anomalous is important. The labelling process enriches the feature data with information such as (i) the type and structure of malicious or anomalous data, and (ii) dependencies among different isolated malicious activities. The correlation and labelling of each feature traffic as normal or anomalous is made using Algorithm 1. However, both normal and anomalous traffics are collected separately in several sessions within a week. We remove normal traffic from anomalous traces as much as possible.

The overall traffic composition with protocol distribution in the generated datasets

Table 4.13: List of features in the KDDcup99 intrusion dataset

Label/feature name	Type	Description
Basic features		
1 Duration	C	Length (number of seconds) of the connection
2 Protocol-type	D	Type of protocol, e g , tcp, udp, etc
3 Service	D	Network service at the destination, e g , http, telnet, etc
4 Flag	D	Normal or error status of the connection
5 Src-bytes	C	Number of data bytes from source to destination
6 Dst-bytes	C	Number of data bytes from destination to source
7 Land	D	1 if connection is from/to the same host/port, 0 otherwise
8 Wrong-fragment	C	Number of "wrong" fragments
9 Urgen	C	Number of urgent packets
Content-based features		
10 Hot	C	Number of "hot" indicators (hot number of directory accesses, create and execute program)
11 Num-failed-logins	C	Number of failed login attempts
12 Logged-in	D	1 if successfully logged-in, 0 otherwise
13 Num-compromised	C	Number of "compromised' conditions (compromised condition number of file/path not found errors and jumping commands)
14 Root-shell	D	1 if root-shell is obtained, 0 otherwise
15 Su-attempted	D	1 if "su root" command attempted, 0 otherwise
16 Num-root	C	Number of "root" accesses
17 Num-file-creations	C	Number of file creation operations
18 Num-shells	C	Number of shell prompts
19 Num-access-files	C	Number of operations on access control files
20 Num-outbound-cmds	C	Number of outbound commands in an ftp session
21 Is-host-login	D	1 if login belongs to the "hot" list 0 otherwise
22 Is-guest-login	D	1 if the login is a "guest" login, 0 otherwise
Time-based features		
23 Count	C	Number of connections to the same host as the current connection in the past 2 seconds
24 Srv-count	C	Number of connections to the same service as the current connection in the past 2 seconds (same-host connections)
25 Serror-rate	C	% of connections that have "SYN" errors (same-host connections)
26 Srv-serror-rate	C	% of connections that have "SYN" errors (same-service connections)
27 Rerror-rate	C	% of connections that have "REJ" errors (same-host connections)
28 Srv-rerror-rate	C	% of connections that have "REJ" errors (same-service connections)
29 Same-srv-rate	C	% of connections to the same service (same-host connections)
30 Diff-srv-rate	C	% of connections to different services (same-host connections)
31 Srv-diff-host-rate	C	% of connections to different hosts (same-service connections)
Connection-based features		
32 Dst-host-count	C	Count of destination hosts
33 Dst-host-srv-count	C	Srv_count for destination host
34 Dst-host-same-srv-rate	C	Same_srv_rate for destination host
35 Dst-host-diff-srv-rate	C	Diff_srv_rate for destination host
36 Dst-host-same-src-port-rate	C	Same_src_port_rate for destination host
37 Dst-host-srv-diff-host-rate	C	Diff_host_rate for destination host
38 Dst-host-serror-rate	C	Serror_rate for destination host
39 Dst-host-srv-serror-rate	C	Srv_serror_rate for destination host
40 Dst-host-rerror-rate	C	Rerror_rate for destination host
41 Dst-host-srv-rerror-rate	C	Srv_rerror_rate for destination host
C-Continuous, D-Discrete		

is summarized in Table 4 14 The traffic includes the TUIDS intrusion dataset, the TUIDS coordinated scan dataset and the TUIDS DDoS dataset. The final labelled feature datasets for each category with the distribution of normal and attack information are summarized in Table 4.15. All datasets are prepared at both packet and flow levels and are presented in terms of training and testing in Table 4 15.

Algorithm 1 . FC and labelling ($\mathbb{F}$)

Input: extracted feature set, $\mathbb{F} = \{\alpha_1, \beta_1\ \gamma_1.\delta_1\}$
Output: correlated and labelled feature data, $\mathbb{X}$

1 initialize X
2. call $FeatureExtraction()$, $F \leftarrow \{\alpha_1, \beta_1, \gamma_1, \delta_1\}$, ▷ the procedure FeatureExtraction() extracts the features separately for all cases
3 **for** $i \leftarrow 1$ to $|\mathbb{N}|$ **do** ▷ $\mathbb{N}$ is the total traffic instances
4 **for** $i \leftarrow 1$ to $|\mathbb{F}|$ **do** ▷ $\mathbb{F}$ is the total traffic features
5· **if** $(unique(src.ip \wedge dst\ ip))$ **then**
6 store $X[ij] \leftarrow \alpha_{1(ij)}, \beta_{1(ij)}$
7 **end if**
8 **if** $((\mathbb{T} == 5s) \wedge (LnP == 100))$ **then** ▷ $\mathbb{T}$ is the time window, LnP is the last n packets
9 Store $X[ij] \leftarrow \gamma_{1(ij)}, \delta_{1(ij)}$
10· **end if**
11. **end for**
12. $X[ij] \leftarrow \{normal, attack\}$ ▷ label each traffic feature instance based on the duration of the collected traffic
13 **end for**

Table 4.14: TUIDS dataset traffic composition

Protocol	*Size (MB)*	*(%)*
(a) Total traffic composition		
IP	66784 29	99 99
ARP	3 96	0 005
IPv6	0 00	0 00
IPX	0 00	0 00
STP	0 00	0 00
Other	0 00	0 00
(b) TCP/UDP/ICMP traffic composition		
TCP	49049 29	73 44
UDP	14940 53	22 37
ICMP	2798 43	4 19
ICMPv6	0 00	0 00
Other	0 00	0 00

4.3.7 Comparison with Other Public Datasets

Several real network traffic traces are readily available to the research community as reported in Section 2. Although these traffic traces are invaluable to the research community most if not all, fail to satisfy one or more requirements described in Section 1. This thesis is mostly distinguished by the fact that the issue of data generation is approached from what other datasets have been unable to provide for the network security community. It attempts to resolve the issues seen in other datasets by presenting a systematic approach to generate real-life network intrusion datasets. Table 4.16 summarizes a comparison between the prior datasets and the

Table 4.15: Distribution of normal and attack connection instances in real-life packet and flow level TUIDS datasets

Connection type	Dataset type			
	Training dataset		Testing dataset	
(a) TUIDS intrusion 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	-
(b) TUIDS coordinated scan dataset				
Packet level				
Normal	65285	90 14%	41095	84 95%
Probe	7140	9 86%	7283	15 05%
Total	72425	-	48378	-
Flow level				
Normal	20180	73 44%	15853	65 52%
Probe	7297	26 56%	8357	34 52%
Total	27477	-	24210	-
(c) TUIDS DDoS dataset				
Packet level				
Normal	46513	68 62%	44328	60 50%
Flooding attacks	21273	31 38%	28936	39 49%
Total	67786	-	73264	-
Flow level				
Normal	27411	57 67%	28841	61 38%
Flooding attacks	20117	42 33%	18150	38 62%
Total	47528	-	46991	-

dataset generated through the application of our systematic approach to fulfill the principal objectives outlined for qualifying datasets

Table 4.16: Comparison of existing datasets and their characteristics

Dataset	*u*	*v*	*w*	*No of instances*	*No of attributes*	*x*	*y*	*z*	*Some references*
Synthetic	No	No	Yes	user dependent	user dependent	Not known	any	user dependent	[5 159]
KDDcup99	Yes	No	Yes	805050	41	BCTW	P	C_1	[63 155, 163,168]
NSL-KDD	Yes	No	Yes	148517	41	BCTW	P	C_1	[248]
DARPA 2000	Yes	No	No	Huge	Not known	Raw	Raw	C_2	[259]
DEFCON	No	No	No	Huge	Not known	Raw	P	C_2	[259]
CAIDA	Yes	Yes	No	Huge	Not known	Raw	P	C_1	[259]
LBNL	Yes	Yes	No	Huge	Not known	Raw	P	C_2	[265]
ISCX-UNB	Yes	Yes	Yes	Huge	Not known	Raw	P	A	[259]
KU	Yes	Yes	No	Huge	24	BTW	P	C_1	[29]
TUIDS	Yes	Yes	Yes	Huge	50 24	BCTW	P F	C_1	[5 159]

u-realistic network configuration
v-indicates realistic traffic
w-describes the label information
x-types of features extracted as basic features (B) content based features (C), time based features(T) and window based features(W)
y-explains the types of data as packet based (P) or flow based (F) or hybrid (H) or others (O)
z-represents the attack category as C_1-all attacks, C_2-denial of service, C_3-probe, C_4-user to root, C_5-remote to local, and A-application layer attacks

Most datasets are unlabelled as labelling is laborious and requires a comprehen-

sive search to tag anomalous traffic. Although an IDS helps by reducing the work, there is no guarantee that all anomalous activity is labelled This has been a major issue with all datasets and one of the reasons behind the post-insertion of attack traffic in the DARPA 1999 dataset, so that anomalous traffic can be labelled in a deterministic manner. Having seen the inconsistencies produced by traffic merging, this chapter has adopted a different approach to provide the same level of deterministic behavior with respect to anomalous traffic by conducting anomalous activity within the capturing period using available network resources. Through the use of logging, all ill-intended activity can be effectively labeled.

The extent and scope of network traffic capture become relevant in situations where the information contained in the traces may breach the privacy of individuals or organizations. In order to prevent privacy issues, almost all publicly available datasets remove any identifying information such as payload, protocol, destination, and flags. In addition, the data is anonymized where necessary header information is cropped or flows are just summarized.

In addition to anomalous traffic, traces must contain background traffic. Most captured datasets have little control over the anomalous activities included in the traces. However, a major concern with evaluating anomaly based detection approaches is the requirement that anomalous traffic must be present on a certain scale. Anomalous traffic also tends to become outdated with the introduction of more sophisticated attacks So, we have generated more up-to-date datasets that reflect the current trends and are tailored to evaluate certain characteristics of detection mechanisms which are unique to themselves.

4.4 Observations and Summary

Several questions may be raised with respect to what constitutes a perfect dataset when dealing with the dataset generation task. These include qualities of normal, anomalous, or realistic traffic included in the dataset. We provide a path and a template to generate a dataset that simultaneously exhibits the appropriate levels of normality, anomalousness, and realism while avoiding the various weak points of currently available datasets, pointed out earlier. Quantitative measurements can

be obtained only when specific methods are applied to the dataset.

The following are the major observations and requirements when generating an unbiased real-life dataset for intrusion detection.

- The dataset should not exhibit any unintended property in both normal and anomalous traffic.
- The dataset should be labeled properly.
- The dataset should cover all possible current network scenarios.
- The dataset should be entirely non-anonymized.
- In most benchmark datasets, the two basic assumptions described in Section 1 are valid but this bias should be avoided as much as possible.
- Several datasets lack traffic features, although it is important to extract traffic features with their relevancy for a particular attack.

Despite the enormous efforts needed to create unbiased datasets, there will always be deficiencies in any one particular dataset. Therefore, it is very important to generate dynamic datasets which not only reflect the traffic compositions and intrusions types of the time, but are also modifiable, extensible, and reproducible. Therefore, new datasets must be generated from time to time for the purpose of analysis, testing, and evaluation of network intrusion detection methods and systems from multiple perspectives.

In this chapter, we have discussed a systematic approach to generate real-life network intrusion datasets using both packet and flow level traffic information. Three different categories of datasets have been generated using the TUIDS testbed. They are (i) TUIDS Intrusion Dataset, (ii) TUIDS Coordinated Scan Dataset, and (iii) TUIDS DDoS Dataset. We incorporate maximum number of possible attacks and scenarios during the generation of the datasets in our testbed network. These datasets are used to evaluate the performance of methods developed for intrusion detection reported in subsequent chapters.

Chapter 5

Outlier-based Approach for Coordinated Port Scan Detection

This chapter presents an overview of port scans, significance of port scans, and the possibilities for detecting them at firewall level. We also discuss several port scan detection methods with a general comparison We introduce an outlier based approach to detect coordinated scans as early as possible. We also proposed an outlier score function to test each candidate object to identify coordinated port scan using score values The method reports each candidate object as normal or coordinated port scan w.r.t. a threshold. This work is evaluated using a real-life coordinated scan datable prepared by us and publicly available probe datasets.

5.1 Introduction

During the last several decades, network defenders and researchers have developed approaches to detect malicious scans as well as coordinated port scans to keep enterprise networks secure. This is because cyber threats are becoming more sophisticated and more numerous, leading to more substantial damages to systems within short periods of time [266, 267]. Two types of correlations are used in a coordinated scan attack, viz., *action correlation* and *task correlation* [268, 269]. How actions performed by one user affects another user is obtained during action correlation. For example, a particular action performed by one user may facilitate another user who performs the actual attack. In the other type of correlation, tasks divided

among multiple users are discovered. Here we focus mainly on task correlation.

Network administrators or defenders are interested in detecting coordinated scan attacks for a system in an enterprise network due to the following reasons.

- To detect coordinated scan attacks just like the detection of other attacks,
- To foil greater interest by the attacker who wants to remain undetected.
- To obviate the potential seriousness of the actual attacks.

A coordinated port scan is a part of a coordinated attack. Here, tasks are distributed among multiple hosts for individual actions which may be synchronized. A port scan is an information gathering method used by an opponent to gain information about responding computers and open ports on a target network host. An opponent initiates the exploration of multiple hosts to scan a portion of the target network, with multiple sources focused on the portion of the target network which they want to compromise after getting relevant information from the target host. Intrusion Detection Systems (IDSs) are normally configured to recognize and report single source port scan activity. So, they cannot usually detect multiple source scans that collaborate with several hosts during scanning

5.1.1 Motivation and Contributions

Early detection of port scans, particularly stealthy or coordinated port scans, is important to enable action against potential intruders. The attackers or intruders are technically sophisticated enough to remain undetected while gathering information but the network defenders are usually out in the open Single source scan detection is comparatively easy to detect because detection usually works better when a single source communicates with a single or multiple destinations. But the detection of a coordinated port scan is difficult due to the lack of relevant feature information at both packet and flow levels. Therefore, we develop an adaptive outlier based detection mechanism for coordinated port scans known as AOCD. This chapter makes the following key contributions.

- We present a survey of existing work on port scan attack detection significantly expanding the discussion in several directions. This also includes overview of

port scans and types of such scans, firewall level detection possibility, detection methods, evaluation and deployment.

- We formalize the problem of coordinated scan detection as a data mining problem and present an approach to transform network traffic data into a form where a classifier can be directly used. Specifically, we select random samples from the dataset and identify a set of features relevant for cluster detection for early detection of coordinated port scans.
- We introduce an outlier score function to test each candidate object to identify coordinated port scan using the estimated score values. The method reports each candidate object as normal or coordinated port scan with respect to a threshold.
- We present extensive experiments using real-world network traffic data. The results show that our approach, which we call AOCD has substantially better performance than other state-of-the-art approaches in terms of accuracy and false positive rate.

5.2 Port Scans and Related Concepts

We present here some preliminary discussions on port scans, types and coordinated port scans.

5.2.1 Port Scans and Types

There are several forms of reconnaissance activity, which often precedes an attack. When an adversary uses an effective mechanism to remotely probe a network, it is known as *port scanning*. System administrators and other network defenders also use this mechanism to detect port scans as precursors to serious attacks [5]. A port scan can be defined as sending packets to a particular IP or port to get a response from an active host in the network indicating the services it offers. A port scan is useful to an attacker who wants to gain substantial information about the target host. Thus, it is of considerable interest to attackers to determine whether or not the defenders of a network scan ports regularly. Attackers hide their identity during

port scanning whereas network defenders do not. Vivo et al. [270] describe a port scan as being composed of hostile Internet searches for open "doors" or "ports", through which intruders gain access to computers Generally, there are several hosts available on a network and they run many services that commonly use TCP or UDP ports for communication with each other These techniques consist of sending a message to a port and listening for an answer The received response indicates port status and can be helpful in determining a host's operating system and other information relevant to launching a future attack A vulnerability scan is similar, except that a positive response from the target results in further communication to determine whether the target is vulnerable to a particular exploit Most attacks are preceded by some form of scanning activity. particularly vulnerability scanning [271].

A computer contains 65536 standardly defined ports [272]. They can be classified into three large ranges: (a) well known ports ($0 - 1023$), (b) registered ports ($1024 - 49151$) and (c) dynamic and/or private ports ($49152 - 65535$). Normally, a port scan helps the attacker in finding those ports that are available to launch attacks, but it does not directly harm the system Essentially, a port scan sends a packet with a message to the target host one at a time and listens for an answer. The response indicates whether the port is being used. This is a probe for weaknesses to launch future attacks TCP and UDP ports are usually used for port scanning but only TCP port scanning returns good feedback to the attacker because it is a connection-oriented protocol. UDP port scanning may not readily give relevant information to the attacker because it is a connectionless protocol In addition, a UDP port may be easily blocked by network defenders or network administrators. The following are the various types of port scans [5] which are used to probe weaknesses from a networked host (shown in Figure 5.1).

- *Stealth scan.* Auditing tools cannot detect this type of scanning because of complicated design architectures. Such a scan sends TCP packets to the destination host with stealth flags Some of the flags are SYN, FIN and NULL.

- *SOCKS port probe*: It allows sharing of Internet connections on multiple hosts.

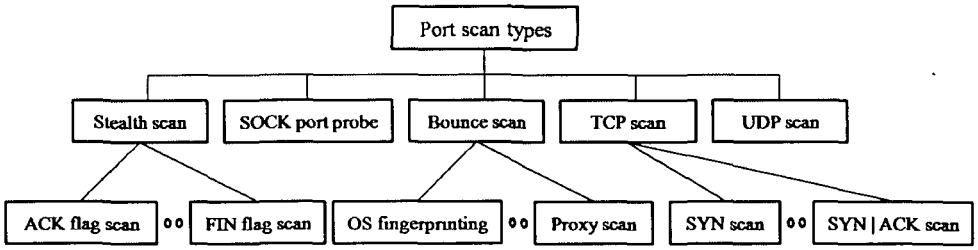


Figure 5.1: Types of port scans

Attackers scan these ports because a large percentage of users misconfigure SOCKS ports, potentially permitting arbitrarily chosen sources and destinations to communicate It also allows the attackers to access other Internet hosts while hiding their true locations.

- *Bounce scan*: An FTP bounce scan attack takes advantage of a vulnerability of the FTP protocol itself. Email servers and HTTP Proxies are the common applications that allow bounce scans.

- *TCP scan*: This type of scanning is used by a smart attacker because it never establishes a connection permanently. The attacker can launch an attack immediately if a remote port accepts the connection request. Normally, this type of connection request cannot be logged by a server's logging system due to its smart connection attempt. Some TCP scans are TCP Connect(), reverse identification, Internet protocol (IP) header dump scan, SYN, FIN, ACK, XMAS, NULL and TCP fragment.

- *UDP scan*: A UDP scan attempts to discover open ports related to the UDP protocol. However, UDP is a connectionless protocol and, thus, it is not often used by attackers since it can be easily blocked.

The list of port scan types discussed above along with firewall detection possibilities during the scanning process are given in Table 5.1. We can see from the table that most of scans are not detected at firewall level.

5.2.2 Coordinated Port Scan

A coordinated port scan is composed of multiple scans from multiple sources where there is a single instigator behind the set of sources. The task of distributed infor-

Table 5.1: Port scan types and their firewall level detection possibilities

Port scanning technique	Protocol	TCP flag	Target reply (open port)	Target reply (closed port)	Firewall level detection possibility
TCP *Connect()*	TCP	SYN	ACK	RST	Yes
Reverse Ident	TCP	No	No	No	No
SYN Scan	TCP	SYN	ACK	RST	Yes
IP Header Dump Scan	TCP	No	No	No	No
SYN|ACK Scan	TCP	SYN|ACK	RST	RST	Yes
FIN Scan	TCP	FIN	No	RST	No
ACK Scan	TCP	ACK	No	RST	No
NULL Scan	TCP	No	No	RST	No
XMAS Scan	TCP	All flags	No	RST	No
TCP Fragment	TCP	No	No	No	No
UDP Scan	UDP	No	No	Port Unreachable	No
FTP Bounce Scan	FTP	Arbitrary Flag Set	No	No	No
Ping Scan	ICMP	No	Echo Reply	No	Yes
List Scan	TCP	No	No	No	No
Protocol Scan	IP	No	-	-	No
TCP window scan	TCP	ACK	RST	RST	No

mation gathering is accomplished using either a many-to-one or a many-to-many model [273,274] The attacker uses multiple hosts to execute information-gathering techniques in two ways: rate-limited, and random or non-linear. In a rate-limited information-gathering technique, the number of packets sent by a host to scan is limited [5,275,276]. This is based on the Berkeley Software Distribution (FreeBSD) implementation of UNIX where separate rate limits are maintained for open ports as well as closed ports. For example, TCP RST is rate limited. "ICMP port unreachable" is also rate limited. On the other hand, a random or non-linear gathering technique refers to randomization of the destination IP-port pairs among the sources, as well as randomization of the time delay for each probe packet. A coordinated attack has a more generic form of a distributed scan than the ones described by Staniford-Chen et al. [277]. It is defined as multi-step exploitation using parallel sessions with the objective of obscuring the unified nature of the attack, allowing the attackers to proceed more quickly. We present the TUIDS testbed architecture (see Figure 5.9) for the generation of coordinated port scans with configuration details.

5.3 Related Research

Based on how scanning is performed, port scan techniques can be classified into two broad categories: *single-source* port scans and *distributed* port scans Each of

these categories is illustrated in Figure 5.2. Therefore, we classify various port scan detection approaches available in the literature into two different categories: *single-source* and *distributed* approaches. Single-source port scan is performed following either a *one-to-one* or a *one-to-many* model for gathering information about a target computer or network. On the other hand, distributed information gathering [273] is performed using a *many-to-one* or *many-to-many* model for gathering information about a target computer or network. A hierarchy of the scan detection approaches is reported in Figure 5.3.

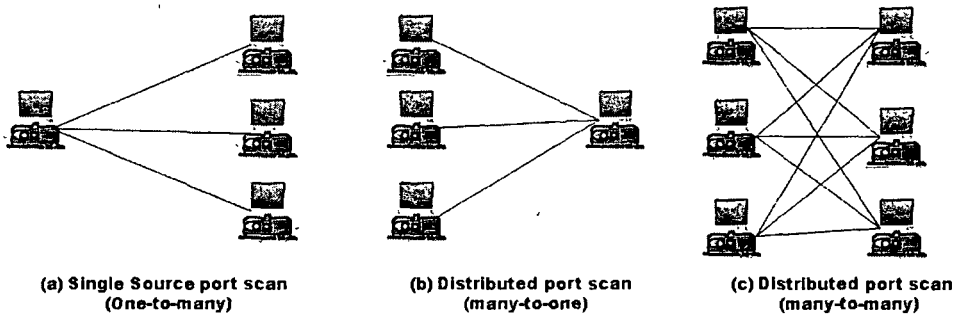


Figure 5.2: Single-source and Distributed port scans

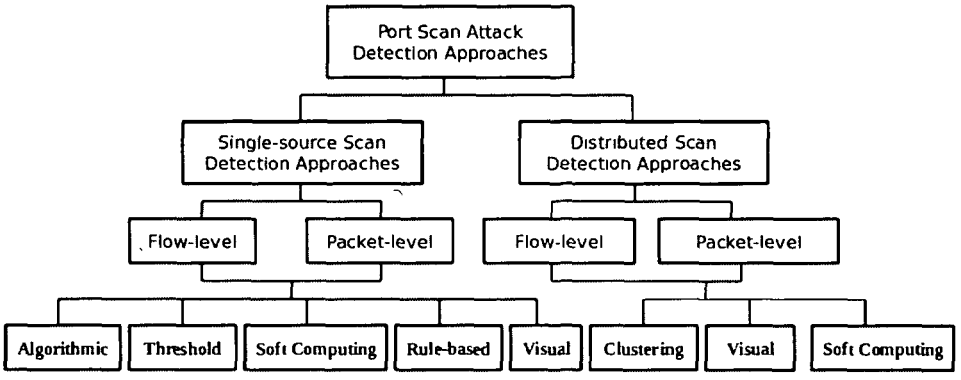


Figure 5.3: Hierarchy of port scan attack detection approaches

5.3.1 Single Source Port Scans and Approaches for Detection

The goal of port scanning from the perspective of an attacker is to gather ideas regarding where to probe for weaknesses. One can scan the network in a *one-to-many* fashion. As discussed in [278], a scan or any network attack can be detected by using a network intrusion detection system (NIDS). In the literature, a port scanner

is defined as consisting of *"specialized programs used to determine what TCP ports of a host have processes listening on them for possible connections"* [270].

Staniford et al. [279] further define scan footprint as the set of ports or IP combinations that the attacker is interested in characterizing. According to them, port scans can be of four types (as shown in Figure 5.4): *vertical*, *horizontal*, *strobe* and *block*. A *vertical* scan consists of a port scan of some or all ports on a single computer. The other three types of scans are used over multiple IP addresses. A

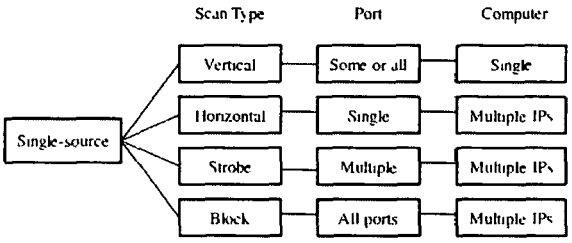


Figure 5.4: Single-source scan types with its ports detail

horizontal scan is a scan of a single port across multiple IP addresses. If the port scan is of multiple ports across multiple IP addresses, it is called a *strobe* scan. A *block* scan is a port scan against all ports on multiple IP addresses. Yegneswaran et al. [280] quantified vertical and horizontal scans, defining a vertical scan as consisting of six or more ports on a single computer, and a horizontal scan as consisting of five or more IP addresses within a subnet.

Detection approaches for single-source port scans have been part of intrusion detection systems since 1990, from the release of Network Security Monitor (NSM) [281]. We divide these detection approaches into five categories. *algorithmic*, *threshold based*, *soft computing*, *rule based*, and *visual* Each of these can be further categorized based on the type of network data processed, methodology used for detection and evaluation criteria. For example, some approaches exploit packet level information whereas some others use flow level information. These details provide not only the connection information, but also allow one to analyze the packet payload. This allows signatures of known attacks to be used on the data to determine whether or not the packet payload contains an attack. Flow level information is provided by Cisco NetFlow [282] and Argus [283] in the form of summarized connection information.

Algorithmic Approaches

These approaches use methods such as hypothesis testing and probabilistic models, to detect port scan attacks based on analysis of network activity. Some of the most well known approaches are discussed below.

Leckie and Kotagiri [284] present an algorithm based on a probabilistic model. For each IP address in the monitored network, the algorithm generates a probability $P(d|s)$ that represents how likely it is that a source will contact that particular destination IP, where d is the destination IP and s is the source, based on how commonly that destination IP is contacted by other sources, $P(d)$. Similarly, it also computes a probability for each port that represents how likely a source will contact a particular destination port, $P(p|s)$ where p is the destination port. A limitation of this approach is that $P(d)$ is based on the prior distribution of sources that have accessed that IP address. This implies that if the probabilities for this approach are generated based on a sample of network data, and if the monitored network is scanned, the resulting distributions may include scans as well as normal traffic. Another limitation of this approach is that it assumes that an attacker accesses the destinations at random; this may not be always true. Kim et al. [285] aim to detect network port scans using anomaly detection. First, the method performs statistical tests to analyze traffic rates. Then, it makes use of two dynamic *chi-square* tests to detect anomalous packets. It models network traffic as a marked point process and introduces a general port scan model. The authors present simulation results to detect 10 malicious vertical scans with true positive rate greater than 90% and false positive rate smaller than 15% for both the static and dynamic tests using the port scan model and statistical tests.

Ertoz et al. [286] develop a system called MINDS (Minnesota INtrusion Detection System) that can analyze network traffic and can also detect port scan attacks. It reads NetFlow data and generates data characteristics, including flow level information, e g., source IP, source port, number of bytes, etc. It then derives information such as the number of connections from a single source, the number of connections to a single destination, the number of connections from a single source to the same port, and the number of connections from a single destination to the

same source port. These four features are counted over a time window and over a connection window. An anomaly score [37] is estimated based on the flow data and derived data for each network traffic record A report ordered by anomaly score is generated The authors also claim that it can detect both fast and slow scanning.

Gates et al. [287] analyze Cisco NetFlow data for port scan attacks. The method extracts the events (bursts of network activities surrounded by quiescent periods) for each source and the flows in each event are then sorted according to destination IP and destination port. It attempts to calculate six characteristics for each event based on statistical analysis of port scans It estimates a probability using logistic regression with these six characteristics as input variables to predict whether the events contain a scan or not. The main drawback of the method is that it is non-real time. Udhayan et al. [288] report a heuristic approach for detecting port scan attacks. One possible solution to curb a zombie army or a malicious botnet attack is by detecting and blocking or dropping reconnaissance scans, i.e., port scans. They derive a set of heuristics to detect these scans, some quite crafty. It is written into the firewall and is triggered immediately after a port scan is detected, to drop packets with the IP address of the source of port scan for a pre-determined period. This detection approach is more user friendly than other approaches like SNORT [161]

Gyorgy et al. [289] propose a model known as off-the-shelf classifier based on data mining. Initially, it transforms network trace data into a feature dataset with label information. Then, it selects Ripper, a fast rule based classifier, which is capable of learning rules from multi-model datasets, with results that are easy to interpret. The authors successfully demonstrate that data mining models can encapsulate expert knowledge to create an adaptive algorithm that can substantially outperform the state-of-the-art for heuristic based scan detection in both precision and recall. This technique is also capable of detecting the scanners at an early stage. Treurniet [290] introduces a new scan detection technique that improves the understanding of Internet traffic The author creates a session model using the behavior of packet-level data between host pairs identified and activities. In a dataset collected over 24 hours 78% of the instances were identified as reconnaissance activities, out of which 80% were slower scans Thus, the method demonstrates its

understanding of Internet traffic by classifying known activities, reporting visible threats to the network through scan detection.

Threshold-based Approaches

These approaches examine events of interest X across a Y-sized time window to detect port scan attacks above certain thresholds [291]. The most commonly used parameter for detecting scans is the number of unique IP addresses contacted by a host. Several intrusion detection systems have been developed in the past couple of years in the public domain that use the threshold-based approach to detect anomaly. The approach requires the packet level information.

Heberlein et al. [281] present a system known as Network Security Monitor (NSM), which is designed using the algorithmic approach and is considered to have pioneered the implementation of threshold based scan detection [269]. This tool has three parts: data capturing, data analysis and support. The data analysis is the core part of the NSM. It collects data in different forms such as statistical, session, full content and alert. Statistical data represent the aggregation of network traffics, protocol breakdown and distribution. Session data represent the connection pairs, and conversation between two hosts. Full content data represent the log of every single bit of network traffic. Alert data represent the data collected by an IDS. It recognizes a source as anomalous and potentially malicious if it is found to contact more than 15 other IP addresses during an unspecified period of time. It also identifies a source as anomalous if it tries to contact an IP address that does not contain a responding computer on the monitored network. With this last heuristic, it assumes that an external source would contact an internal IP address only for a reason backed by knowledge of the existence of a service at an internal IP address such as an FTP server or a mail server. NSM is neither a security event management system nor an intrusion prevention system. Roesch [161] presents a signature-based intrusion detection system known as SNORT. It uses a pre-processor that extracts port scans, based on either invalid flag combination (for example, NULL scans, Xmas scans, and SYN-FIN scans) or on exceeding a threshold. SNORT uses a pre-processor, called *portscan* that watches connections to determine whether a scan is occurring. By default, SNORT is configured to generate an alarm only if it has

detected SYN packets sent to at least five different IP addresses within 60 seconds or 20 different ports within 60 seconds, although this can be adjusted manually. By having such a high threshold, the number of false positives is reduced. However, a careful scan at a rate lower than the threshold can easily go undetected.

Paxson [292] introduces a detection system known as *Bro* that attempts to detect scans based on a thresholding approach Network scans are detected when a single source contacts multiple destinations (> some threshold). It also detects vertical scans when a single source contacts too many different ports. It assumes that the external site has initiated the conversation in both cases However, a major limitation of this method is the increased number of false positives. Bro uses payload as well as packet level information Jung et al. [269] describe an approach called Threshold Random Walk (TRW) based on sequential hypothesis testing It detects port scans using an Oracle database that contains the assigned IP addresses and ports inside a network after performing an analysis of return traffic. When a connection request is received, the source IP is entered into a list, along with each destination to which this source has attempted a connection. If the current connection is to a destination which is already in the list, the connection is ignored. If it is to a new destination, it is added to the list, and a measure that determines whether the connection is scanning or not is computed and updated based on the status of the connection. The entire source is flagged as either scanning or not scanning depending on whether the measure has exceeded the maximum threshold or has dropped below the minimum threshold, respectively. It has been observed that benign activity rarely results in connections to hosts or services that are not available, whereas scanning activity often makes such connections, with the probability of connecting to a legitimate service dependent on the density of the target network.

Romig [293] develops a flow analysis tool called *flow-dscan*. This tool examines flows for floods and port scans. Floods are identified by an excessive number of packets per flow. Port scans are identified by a source IP address contacting more than a certain threshold number of destination IP addresses or destination ports (only ports less than 1024 are examined) on a single IP address. To minimize the false alarm rate, this approach makes use of a suppress list consisting of IP addresses.

Zhang and Fang [294] propose a new port scan detection approach known as Time-based Flow size Distribution Sequential hypothesis testing (TFDS) for high-speed transit networks where only unidirectional flow information is available TFDS uses the main ideas of sequential hypothesis testing to detect scanners that exhibit abnormal access patterns in terms of flow size distribution (FSD) entropy This work makes a comparison with the state-of-the-art backbone port scan detection method TAPS [295] in terms of efficiency and effectiveness using real backbone packet trace, and finds that TFDS performs much better than TAPS

Gadge and Patil [296] propose a method to identify possible port scans and try to gather additional information about the scanner or attacker, such as probable location and operating system The scan detection system collects all the information and stores it to generate reports in terms bar graphs Analysis of stored data can be done in terms of time and day by which type of scan was performed from which IP the scan was performed different ports, etc Based on the analysis of the various parameters used, it can recognize and report the type of attack or scan performed during a time window This method can detect scans coming from most common scanners such as Angry IP nmap and MegaPing

Soft Computing Approaches

Soft computing includes important methods that provide flexible information processing for handling real-life ambiguous situations [297] Methods in soft computing exploit tolerance for imprecision and uncertainty, use approximate reasoning and partial truth in order to achieve traceability, provide robustness and low-cost solutions to problems Some soft computing approaches for scan detection are discussed next

Chen and Cheng [298] present a novel and fast port scan detection method based on Partheno-Genetic Algorithms (PGA) The method can efficiently discover ports that are open most often During genetic evolution ports with more open times survive to the next generation with higher probabilities This approach demonstrates that PGA-based port scan is efficient for average as well as worst cases Sequential port scans are better in best cases only Liu et al [299] discuss a method known as Naive Bayes Kernel Estimator (NBKE), which is used to identify flooding attacks

and port scans from normal traffic. The method represents all known attacks in terms of traffic features. The method takes hand-identified traffic instances as training examples for the NBKE. This method achieves high accuracy in the detection of flooding attacks and port scan attacks. The authors show that the Kernel-based Estimator can provide improved accuracy of 96.8% over the simple Naive Bayes estimator

Shafiq et al. [300] report a comparative study of three classification schemes for automated port scan detection. These includes a simple fuzzy inference system (FIS) that uses classical inductive learning, a neural network that uses the back propagation algorithm and an adaptive neuro fuzzy inference system (ANFIS) that also employs the back propagation algorithm. They use two information theoretic features, namely entropy and KL-divergence of port usage, to model network traffic behavior for normal user applications The authors carry out an unbiased evaluation of these schemes using an endpoint based traffic dataset. This work shows that ANFIS, though more complex, successfully combines the benefits of the classical FIS and Neural Network to achieve excellent classification accuracy

Rule-based Approaches

Generally, a rule-based IDS analyzes traffic data passing through it and differentiates intrusive traffic behaviors from the normal. A rule-based IDS uses rules stored in its knowledge base to detect and take actions when anomaly occurs in the traffic or when there are unauthorized activities. A rule-based IDS must generate rules based on network activity for detecting anomaly. Some rule-based approaches are described below.

Mahoney and Chan [182] introduce a system known as Packet Header Anomaly Detection (PHAD) that learns the normal range of values for all 33 fields in the Ethernet, IP, TCP, UDP, and ICMP headers. A score is assigned to each packet header field in the testing phase and the fields' scores are summed to obtain a packet's aggregate anomaly score. The authors evaluate PHAD using the packet header fields: *source IP*, *destination IP*, *source port*, *destination port*, *protocol type*, and *TCP flags*. Normal intervals for the six fields are learned from 5 days of training data. In the test data, field values not falling in the learned intervals are flagged

as suspect. The top n packet score values are labeled anomalous. The value of n is varied over a range to obtain ROC curves. Another relevant work is proposed by Oke and Loukas [301]. The authors propose a Denial of Service (DoS) detection approach which uses multiple Bayesian classifiers and random neural networks (RNN). Their method is based on measuring various instantaneous and statistical variables describing the incoming network traffic, acquiring a likelihood estimation and fusing the information gathered from the individual input features using likelihood averaging and different architectures of RNNs Kim and Lee [302] suggest an abnormal traffic control framework (ATCF) to detect slow port scan attacks using fuzzy rules. ATCF acts as an intrusion prevention system disallowing suspicious network traffic It manages traffic with a two step policy: (i) decreasing network bandwidth and then (ii) discarding traffic. The authors show that the abnormal traffic control framework can effectively detect slow port scan attacks using fuzzy rules and a stepwise policy

In addition to these two, several other rule-based IDSs have been discussed in the literature that are not included here due to being non-relevant to port scan attack detection.

Visual Approaches

Some approaches present data to the user in a visual manner so that he or she can recognize scans by the patterns it generates. Such approaches detect and investigate port scans using packet level information and flow level information. Some visual approaches are presented here.

Muelder et al. [303, 304] present PortVis, a tool designed for scan detection. It uses summarized network traffic for each protocol and each port for a user-specified time period. The summaries include the number of unique source addresses, the number of unique destination addresses, and the number of unique source-destination address pairs A series of visualization techniques and drill-downs are used to determine whether the monitored traffic contains horizontal or vertical scans. It is unclear how well this algorithm scales to larger networks. It is because this approach requires a manual analysis of the visualizations, rather than an automated recognition of scans Musa and Parish [305] describe prototype

software that enables visualization alerts effectively in real-time. The prototype software incorporates various projections of the alert data in 3-dimensional displays. Filtering, drill-down, and playback of alerts at variable speeds are incorporated to strengthen analysis The developers integrate a false alert classifier using a decision tree algorithm to classify alerts into false and true alerts. The authors also work on the analysis of both *portsweep* and *ntinfoscan* attacks.

Lee et al [306] present an extended version of the NVisionCC system [307], which is a clustering tool based on an extensible visualization framework It exploits the nature of large-scale commodity clusters to improve illegal service detection mechanisms. The cluster properties are only visible when one ceases to look at the cluster as a collection of disparate nodes. The tool can help make insightful observations by correlating open network ports observed on cluster nodes with other emergent properties such as the number and nature of active processes and the contents of important system files. This approach can greatly restrict the actions that an attacker can carry out undetected. ScanViewer [308] is a visual interactive network scan detection system designed to represent traffic activities that reside in network flows and their patterns. ScanViewer combines characteristics of network scans with novel visual structures, and utilizes a set of visual concepts to map the collected datagram to the graphs that emphasize their patterns Additionally, it provides Localport, a tool that captures large-scale port information It has been experimentally shown that ScanViewer not only can detect network scans, port scans, distributed port scans, but also can detect hidden scans. Finally, a graph theoretic model for port scan detection by visualizing graph features for each network connection is reported by [309].

Discussion

A large number of techniques for detection of port scans have been reported in this section under five distinct categories of approaches. However, it is not always easy to unambiguously classify a technique into any one of these approaches since often it uses elements from multiple classes. These approaches use features such as source IP and port, destination IP and port. protocol, start time and end time of the session, and the number of bytes, and packets transferred. Table 5.2 provides

a summary of the scan detection approaches that are available for detecting the single-source port scan attacks. Table 5.2 also shows the performances of those detection techniques wherever available and the datasets used for their evaluation.

Table 5.2: Comparing single-source port scan detection approaches

Detection Approach	Author (s)	Nature of Detection (Real/Non-real time)	Packet(P) / Flow(F) level	Performance	
				False Positive (%)	Detection Rate (%)
Algorithmic	Staniford-Chen et al [277]	R	P	0.03 [284]	93.82 [289]
	Porras and Valdes[310]	N	F		
	Kato et al [311]	R	P		
	Leckie and Kotagiri[284]	R	P		
	Ertoz et al [286]	R	F		
	Kim et al [285]	R	P		
	Gyorgy et al [289]	N	P		
	Gates et al [287]	R	F		
	Udhayan et al [288]	R	P		
	Treurniet [290]	R	P		
Threshold	Heberlein et al [281]	N	P	0.96 [269]	
	Paxson[292]	R	P		
	Roesch[161]	R	P		
	Romig[293]	R	F		
	Jung et al [269]	N	P		
	Gadge and Patil[296]	N	P		
	Zhang and Fang[294]	R	F		
Soft Computing	Streilein et al [312]	R	P	0.1 [312]	100 [312]
	Liu et al [299]	N	P		
	Shafiq et al [300]	N	P		
	Chen and Cheng[298]	N	P		
	Chen and Cheng[298]	N	P		
Rule-based	Mahoney and Chan[182]	N	P		
	Kim and Lee[302]	N	P,F		
Visual	Muelder et al [303]	R	F	0.4 [306]	95.5 [306]
	Abdullah et al [313]	R	F		
	Lee et al [306]	N	P		
	Musa and Parish[305]	R	F		
	Jiawan et al [308]	N	F		
	Cheng et al [309]	N	P		

5.3.2 Distributed Port Scans and Approaches for Detection

Distributed information gathering is performed using either a *many-to-one* or a *many-to-many* model [273]. Here, the attacker uses multiple hosts to execute information gathering techniques in two ways: *rate-limited* and *random* or *non-linear*. In a *rate-limited* information gathering technique, the number of packets sent by a host to scan is limited [276]; this is based on the FreeBSD implementation of Unix where separate rate limits are maintained for open ports as well as closed ports. For example: TCP RST is rate limited, ICMP port unreachable is rate limited, and so on. On the other hand, a *random* or *non-linear* gathering technique refers to randomization of the destination IP-port pairs amongst the sources, as well as

randomization of the time delay for each probe packet.

A *coordinated* attack has a more generic form of a distributed scan described by Staniford-Chen et al. [277]. It is defined as multi-step exploitation using parallel sessions with the objective of obscuring the unified nature of the attack or allowing the attackers to proceed more quickly Green et al [314] define a *coordinated* attack as *"multiple IP addresses working together towards a common goal"*. They also add that a coordinated attack can be viewed as multiple attackers working together to execute a distributed scan on many internal addresses or services. Staniford et al [279] later define a distributed scan as one that is launched from a number of different real IP addresses, so that the scanner can investigate different parts of the footprint from different places. An attacker can scan the Internet using a few dozen to a few thousand zombies. A zombie is a compromised host, whose owner is unaware that the computer is being exploited (a remote attacker has accessed and set up to forward transmissions (spams or viruses) to other computers on the network) by the external party. Yegneswaran et al. [280] define *coordinated* scans as being scans from multiple sources aimed at a particular port of destinations within a one hour window. These scans usually come from more aggressive or active sources that comprise several collaborative peers working in tandem. Finally, Robertson et al. [315] group source addresses together as forming a potentially distributed port scan if they are sufficiently close, where the scanner simply obtains multiple IP addresses from his Internet service provider (ISP). It should be noted that all of these definitions imply some level of co-ordination among the single sources used in the scan

The main goal of these approaches is to detect coordinated attacks. These types of attacks attempt to compromise a single host from multiple systems. There are various methods for detecting these attacks Like the single-source scan detection approaches, the approaches also can be categorized into four classes based on the features used by the methods: algorithmic, clustering, soft computing, and visual.

Algorithmic Approaches

Only a few algorithmic approaches that operate in a distributed mode can be found in the literature. We describe here two popular techniques which perform satisfac-

torily over multiple datasets

Gates [291] describes a model of potential adversaries based on the information they wish to obtain, where each adversary is mapped to a particular scan footprint pattern The adversary model forms the basis of an approach to detect forms of coordinated scans, employing an algorithm that is inspired by heuristics for the set covering problem. The model also provides a framework for comparing various types of adversaries different coordinated scan detection approaches might identify. The author evaluates the model to analyze the detector performance over a set of different datasets Both the detection and false positive rates gathered from the experiments are modeled using regression equations.

Whyte [316] describes the design, implementation and evaluation of fully functional prototypes to detect internal and external scanning activity at an enterprise network. These techniques offer the possibility of identifying local scanning systems within an enterprise network after the observation of only a few scanning attempts with a low false positive and negative rates To detect external scanning activity directed at a network, it makes use of the concept of exposure maps that are identified by passively characterizing the connectivity behavior of internal hosts in a network as they respond to both legitimate connection attempts and scanning attempts The exposure maps technique enables: (1) active response options to be safely focused exclusively on those systems that directly threaten the network, (2) the ability to rapidly characterize and group hosts in a network into different exposure profiles based on the services they offer, and (3) the ability to perform a reconnaissance activity assessment (RAA) that determines what specific information was returned to an adversary as a result of a directed scanning campaign. Finally, the author experimented with real-life scan activity as well as offline datasets. Singh and Chun [317] implement a TCP based port scanner in the OMNeT++ simulator. The authors describe two modules: simple and compound. Both modules are implemented using C++. They claim that their approach can detect TCP connect(), TCP SYN (half-open), TCP FIN (stealth), Xmas, NULL, ACK, Window and Reset (RST) scans at the router level.

Clustering Approaches

Clustering is the process for partitioning data into groups of similar objects. It is an unsupervised learning process There are many approaches available for detecting network scans based on the similarity of the data, compactness of the cluster, and so on. Some approaches are discussed below.

Robertson et al. [315] define distributed port scan as a set of port scans that originate from source IP addresses that are located close together. In other words, they assume that a scanner is likely to use several IP addresses on the same subnet. This implies that if a particular IP address scans a network, IP addresses near this IP address, rather than those far away, are more likely to have also scanned the network Yegneswaran et al. [280] can detect coordinated port scans where a distributed port scan is defined as a set of scans from multiple sources (i.e., five or more) aimed at a particular port at the destinations within a 1-hour window. On the basis of this definition, the authors find that a large proportion of daily scans are coordinated in nature, with coordinated scans being roughly as common as vertical and horizontal scans. The system looks to see if different sources start and stop scanning either at the same time, or in very similar temporal patterns. There is little locality in the IP space for these coordinated scanning sources. The authors do not discuss characteristics of the target hosts.

Staniford et al. [279] present an approach that begins with an analysis of the stealthy port scan detection problem using an intrusion correlation engine. The authors maintain records of event likelihood to estimate the anomalousness of a given packet. For effective detection performance, they use simulated annealing to cluster anomalous packets together into port scans based on heuristics developed from real scans. Packets that score high on anomalousness are kept around longer. They claim that the system is capable of detecting all scans detected by all other current techniques, plus many stealthy scans, with a manageable proportion of false positives.

Soft Computing Approaches

In addition to the approaches discussed so far, there are several distributed scan detection approaches that use soft computing techniques. Next we discuss a few of these.

Curtis et al. [318] describe an intrusion response architecture based on intelligent agents to detect distributed port scans. The authors use a master analysis agent to find a confidence measure based on observed false positive rates. The master analysis agent can combine various alerts using a two-level fuzzy rule set to determine whether a current attack is a continuation of a previous attack, or a new attack. The agent considers characteristics of the attack such as the time between the incident reports, IP addresses, the user name, and the program name. The details of the fuzzy logic employed are not provided in this article, nor are the results of any experiments indicating how well this algorithm performs. Zhang et al. [319] present a distributed multi-layer cooperative model for scan attack detection composed of feature-based detection, scenario-based detection and statistics-based detection. A scan attack always happens at the network layer and the transport layer. The authors categorize scan techniques into three: port scan, bug scan, and detecting scan. The authors claim that the model not only detects common scan attacks or their variants, but can also detect some slow scan attacks, camouflage attacks and DoS attacks that use the TCP/IP protocol.

Visual Approaches

These approaches are used for visualizing network traffic to detect whether the flow of network packet is an attack or normal behavior. One such commonly found approach is proposed by Conti and Abdullah [320]. The approach (discussed in the context of single-source port scan earlier) attempts to detect distributed scans against a background of normal traffic based on visualization. Due to lack of details, it is difficult to understand how a distributed scan would use this tool. Also, it is not clear how much traffic can be viewed at one time without obscuring features of interest. Stockinger et al. [321] present a conditional histogram based detection mechanism for distributed port scans. This method is evaluated using 2.5 billion

Table 5.3: Comparing distributed port scan detection approaches

Detection Approach	Author (s)	Real (R)/ Non-real time (N)	Packet (P)/ Flow (F) level	Performance	
				False Positive Rate (%)	Detection Rate (%)
Algorithmic	Carrie [291]	N	P		
	Whyte [316]	R	P		
	Singh and Chun [317]	N	P		
Clustering	Streilein et al [312]	R	A	0 1 [312]	100 [312] [279]
	Staniford et al [279]				
	Seth et al [315]	N	A	4 [315]	
	Vinod et al [280]	N	A		
Soft Computing	Curtis et al [318]	N	A		80 [318]
	Zhang et al [319]	N	P/F		
Visual	Conti and Kulsoom [320]	R	P		
	Stockinger et al [321]	N	P		
	Baldoni et al [322]	N	P		

network connections with an interactive time interval. Finally, a collaborative architecture to detect coordinated port scans is introduced at [322]. The method aims to identify coordinated attacks with low false alarm rate and accurate separation of group of attackers even they are overlap The method is tested using real network traffic traces.

Discussion

Most distributed port scan detection approaches analyze packet level information. They can detect port scan attacks using IP addresses (source IP, destination IP), connection information, and port fields (source ports, destination ports) in the IP header. A general comparison of the distributed scan detection approaches discussed in this section is given in Table 5.3. We see in column 4 of the table that most of these approaches are non-real time and their performance is evaluated in terms of the false positive rate and the detection rate.

5.4 Problem Statement

Coordinated or distributed port scans originate at multiple sources and focus on a single machine or multiple target machines. It is of special interest to large organizations with high level network situational awareness or military operations to detect coordinated port scans. The following are key problems.

- Coordinated scans compromise the victim machine earlier than single source

port scans.

- Coordinated port scans are distributed in nature. So, a single attacker or intruder coordinates a group of attackers in order to obtain vulnerability information on a set of target networks. This event also consumes the network bandwidth and resources slowly.

To overcome these problems, we develop an adaptive outlier based coordinated port scan detection approach. Coordinated scans can be detected in two different ways, viz., (i) using direct network traffic fields (e.g., source IP, destination IP, protocol, etc.), and (ii) using relevant extracted traffic features (e.g., duration, source bytes, destination bytes, etc.). We follow the second approach to detect coordinated scans. Let $\mathbb{X}$ be the captured, preprocessed current network traffic feature dataset, where $x_1, x_2, \cdots x_s$ are the training samples, randomly selected from dataset $\mathbb{X}$ that contain only normal instances. We apply the fuzzy c-means algorithm to cluster each sample individually into k clusters. Each cluster uses a range based profile for detection. Let $x_1, x_2, \cdots x_t$ be the test instances to classify as attack or normal w.r.t. a threshold δ_2. The profile base is updated if any new distinct instances appear for testing. Thus, our method called AOCD adaptively updates its profile base for the new distinct instances.

5.5 AOCD: The Proposed Approach

We describe the required concepts first and then the AOCD (Adaptive Outlier-based Coordinated scan Detection) algorithm to detect coordinated port scans. The framework for AOCD is given in Figure 5.5.

5.5.1 Outliers and Anomaly Detection

An outlier is an abnormal or infrequent event or object that varies significantly from the normal event or object in terms of a distance measure. A network administrator needs to define an abnormal event based on normal statistics [323]. Outlier detection discovers exceptional events from small or large datasets [324]. Examples of outliers in a two dimensional dataset are illustrated in Figure 5.6

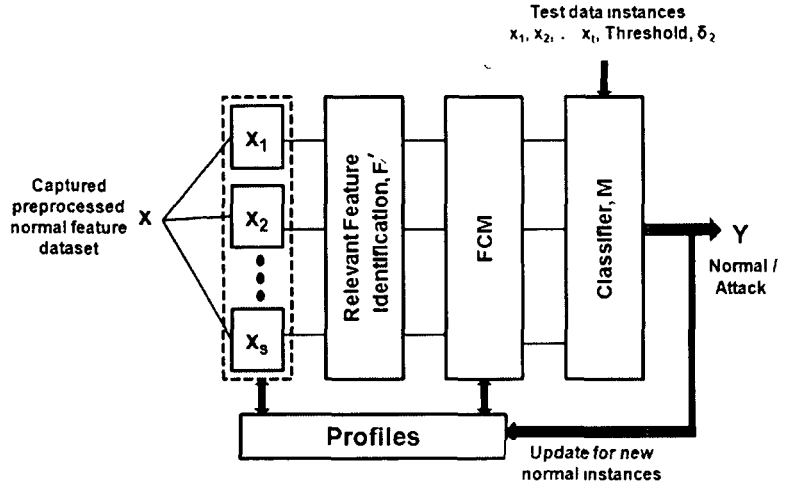


Figure 5.5: A framework for AOCD: FCM is the fuzzy C-means clustering algorithm for sample clustering and F' is the PCA based feature selection technique for each sample as well as testing instances.

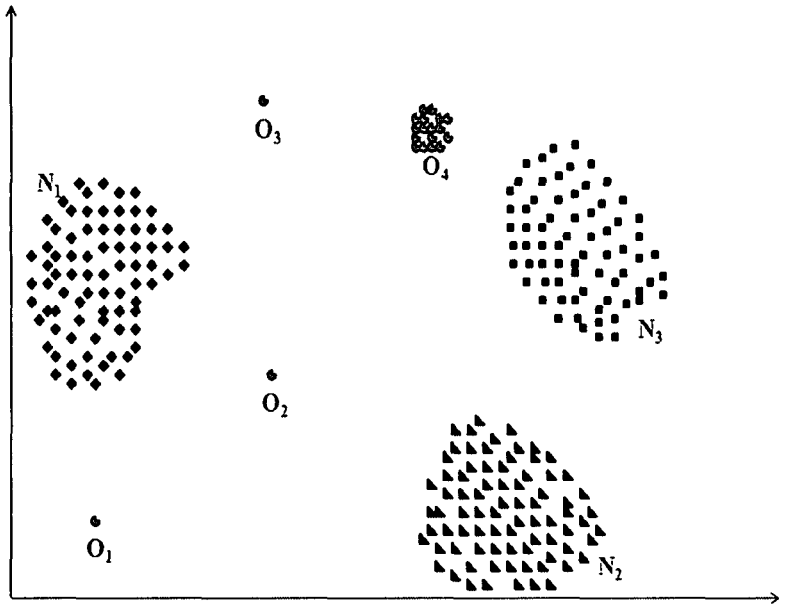


Figure 5.6: Outliers in two dimensional dataset: N_1, N_2, and N_3 are the three normal regions. Points that are sufficiently far away from the normal region (e.g., points O_1, O_2, O_3 and points in O_4 regions) are outliers.

Outlier Score and Its Importance

A large number of outlier detection techniques have been proposed in the literature but only a few of them have been applied to anomaly detection [65, 325]. An outlier score is a summarized value based on distance, density or other statistical measures. A reference based outlier score is presented by Pei and Zaiane [3] for

detecting outliers in large datasets. The authors estimate outlier score based on distance and a degree of nearest neighbor density. The authors define the outlier score as

$$ROS(x) = 1 - \frac{D^P(x,t)}{\max\limits_{1\leq i\leq n} D^P(x_i,t)} \tag{5.1}$$

where $D^P(x,t)$ is $\min\limits_{1\leq r\leq R} D(x,t,p_r)$, and p_r is the closest reference point to P, i.e., $P = \{p_1, p_2, p_3, \cdots p_R\}$, $\mathbb{R}$ is the number of reference points. $D^P(x,t)$ is the degree of neighborhood density of the candidate data point x with respect to the set of reference points P, n is the total number of data points, and t is a reference based nearest neighbor. The $D(x,t,p)$ is the relative degree of density for x in the one dimensional data space x^P and defined as

$$D(x,t,p) = \frac{1}{\frac{1}{t}\sum_{j=1}^{t} |dist(x_j,p) - dist(x,p)|} \tag{5.2}$$

where $dist(x_j,p)$ is the distance between x_j and the reference point p within t^{th} nearest neighbor, $dist(x,p)$ is the distance of x from the reference point p. The candidate data points are ranked according to their relative degrees of density computed with respect to a set of reference points. Outliers are those with high scores. This technique can discover multiple outliers in larger datasets. However, three main limitations of a technique that depends on an outlier score like ROS [3] are the following.

- The score does not always vary with the change of candidate data points.
- Summarizing the data points in terms of scores may not be effective for some attacks.
- It does not work with high dimensional datasets.

To overcome these drawbacks of the outlier score ROS, we have developed an enhanced outlier score function called ROS' presented later in Subsection 5.5.3.

5.5.2 Feature Extraction Using PCA

Principal Components Analysis (PCA) is often used to reduce the number of dimensions in data for cost-sensitive analysis [326]. Let $x_1, x_2, x_3, \cdots x_d$ and $y_1, y_2, y_3, \cdots y_d$

be two d dimensional observations. PCA is concerned with explaining the variance-covariance structure of a set of variables through a few new variables which are functions of the original variables. Principal components are particular linear combinations of the d random variables $x_1, x_2, x_3, \cdots x_d$ with three important properties: (i) The principal components are uncorrelated, (ii) The first principal component has the highest variance, the second principal component has the second highest variance, and so on, and (iii) The total variation in all the principal components combined is equal to the total variation in the original variables $x_1, x_2, x_3, \cdots x_d$. They are easily obtained from an eigen analysis of the covariance matrix or the correlation matrix of $x_1, x_2, x_3, \cdots x_d$.

Let dataset x be denoted as $\{x_1, x_2, x_3 \cdots x_n\}$ with n objects, where each x_i can be a numeric or categorical attribute represented by a d-dimensional vector, i.e., $x = \{x_{i,1}, x_{i,2}, x_{i.3} \cdots x_{i,d}\}$.

Let A be an $n \times d$ covariance matrix of n observations in d dimensional space, i.e., each d random variables $x_1, x_2, x_3, \cdots x_d$. If $(\lambda_1, e_1), (\lambda_2, e_2), (\lambda_3, e_3), \cdots, (\lambda_d, e_d)$ are the d eigenvalue-eigenvector pairs of A, and $\lambda_1 \geq \lambda_2 \geq \lambda_3, \cdots, \lambda_d \geq 0$, then the i^{th} sample principal component of an observation vector, $x = (x_1, x_2, x_3, \cdots x_d)'$ is

$$y_i = e_i' z = [e_{i1}' z_1, e_{i2}' z_2, e_{i3}' z_3, \cdots, e_{id}' z_d] \tag{5.3}$$

where '$'$' represents the transpose of the matrix, $e_i = (e_{i1}, e_{i2}, e_{i3}, \cdots, e_{id})$ is the i^{th} eigenvector and $z = (z_1, z_2, z_3, \cdots, z_d)'$ is the vector of standardized observations defined as $z_k = \frac{x_k - \overline{x_k}}{\sqrt{s_k}}$ where $\overline{x_k}$ and s_k are the sample mean and sample variance of the variable x_k. The features, F' are selected based on the eigenvectors with highest eigenvalues in d dimensional space. Therefore, our approach works on reduced feature spaces given by PCAF, which is based on PCA.

5.5.3 The Proposed Approach

AOCD aims to detect anomalous patterns, i.e., coordinated port scans using an adaptive outlier based approach with reference to profiles. Initially, we select random samples, $x_1, x_2, \cdots x_s$ using a linear congruential generator from the dataset x for training purpose. It is a maximum length pseudo random sequence generator

[327] and can be defined as $x_n = (ax_{n-1} + b)\ mod\ m$, where x_n is the n^{th} number of the sequence, x_{n-1} is the previous number of the sequence. a, b, and m are secrets; a is the multiplier, b is the increment, and m is the period length when m is prime, the maximum period length is $(m-1)$.

We cluster each sample into k classes by using the Fuzzy C-means [328] clustering technique. We obtain the following clusters from all samples: $C_{11}, C_{12}, C_{13}, \cdots C_{1k}$, $C_{21}, C_{22}, C_{23}, \cdots C_{2k}, \cdots\ C_{s1}, C_{s2}, C_{s3}, \cdots C_{sk}$. The method compares a range-based profile for each cluster and matches each profile with others to remove redundancy. These profiles are used as reference during score computation. Finally, the method computes score for each candidate object and reports as normal or outlier (i.e., attack) w.r.t. a threshold, δ_2. We present the Fuzzy C-means clustering technique for cluster formation in Algorithm 2. In Algorithm 2, r is the weighting exponent also known as fuzzifier that influences the performance of clustering. During experimentation, we set $r = 4.5$ for better clustering results.

Algorithm 2 FCM (x, k, r, l_1, ϕ)

Input: x_i is the i^{th} data instance and u_{ij} represents the whole data matrix, k is the number of clusters, r is a real number greater than 1, l_1 is the number of iterations, ϕ is the termination criteria between 0 and 1.

Output: Generate cluster, $C_1, C_2, C_3, \cdots C_k$.

1: Initialize $U = [u_{ij}]$, $U^{(0)}$.

2: Compute the center vectors $k^{(l_1)} = [k_j]$ with $U^{(l_1)}$: $k_j = \frac{\sum_{i=1}^{N} u_{ij}^r x_i}{\sum_{i=1}^{N} u_{ij}^r}$

3: Update $U^{(l_1)}, U^{(l_1+1)}$: $u_{ij} = \frac{1}{\sum_{l_1=1}^{k} \left(\frac{\|x_i - k_j\|}{\|x_i - k_{l_1}\|} \right)^{\frac{2}{r-1}}}$ w.r.t. F'.

4: **if** $\|U^{l_1+1} - U^{l_1}\| < \phi$ **then**

5: Stop.

6: **else**

7: Return to Step 2.

8: **end if**

Let S_i be the number of classes to which each of k' nearest neighbor data objects belongs. k' plays an important role in score computation. Let x_t be a test data object in $\mathbb{X}_t$ and $dist(x_t, R)$ be the distance between the data object x_t and the reference points R, where $t = 1, 2, 3, \cdots n$, $dist$ is a proximity measure used, and $\mathbb{X}_t$ represents the whole candidate dataset. The proposed approach works well with any commonly used proximity measure. The outlier score for a data object x_t we define is given in Equation (5.4).

$$ROS'(x_t) = \frac{\max\limits_{1\le i\le k'} S_i}{k'} \times \left(\frac{\left(1 - \min\limits_{1\le i\le k'} dist(x_t, R_i)\right) \times \left(\sum_{i=1}^{k'} \min\limits_{1\le i\le k'} dist(x_t, R_i)\right)}{\sum_{i=1}^{k'} \max\limits_{1\le i\le k'} dist(x_t, R_i)} \right) \quad (5.4)$$

Here, $\frac{\max\limits_{1\le i\le k'} S_i}{k'}$ is the maximum probability that a data object belongs to a particular class, the remaining part is the summarized value of distance measure within k' nearest neighbors over the relevant feature subset. R_i represents the reference points within k' nearest neighbor. The candidate data objects are ranked based on the outlier score values. Objects with scores higher than a user defined threshold δ_2 are considered as anomalous or outliers. δ_2 is determined by a heuristic method. To test effectiveness, we consider seven different cases (illustrated in Figure 5.7 [5]) and the proposed algorithm is capable of identifying all these seven cases.

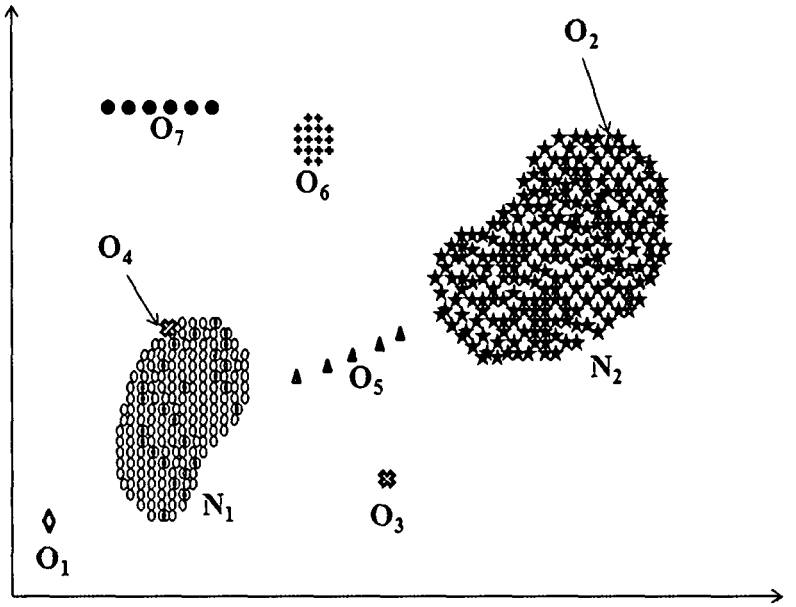


Figure 5.7: Illustration of seven different cases: N_1 and N_2 are two normal clusters; O_1 is the distinct outlier; O_2, the distinct inlier; O_3, the equidistance outlier; O_4, the border inlier; O_5, a chain of outliers; O_6 is another set of outlier objects with higher compactness among the objects; and O_7 is an outlier case of "*stay together*".

A heuristic estimation of k' values for our own flow level dataset for a range of values of accuracy is given in Figure 5.8. We now present a few definitions before we present our algorithm.

Definition 1. ***Pattern Similarity***: Two data objects x_1 and x_2 are defined

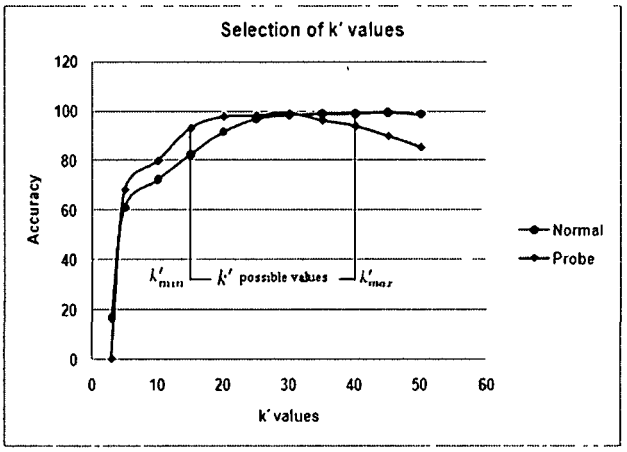


Figure 5.8: k' values vs accuracy in our own flow level dataset.

as similar iff (a) $dist(x_1, x_2) < \alpha'$, and (b) $dist(x_1, x_2) = 0$, if $x_1 = x_2$.

Definition 2. ***Profile***: A profile of a cluster C_i is a range value, $\mu(x_{\mu,1}, x_{\mu,2} \cdots x_{\mu,d})$ of dataset x, where each $x_{\mu,j}$ is the range of the j^{th} column of the respective cluster C_i.

Definition 3. ***Outliers***: Two data objects, O_i and O_j are defined as outliers w.r.t a cluster C_i iff (a) $ROS'(O_i, \mu_i) \geq \delta_2$ where μ_i is the profile of C_i, and (b) for any other data object O_j in C_i, $dist(O_i, O_j) > \delta_2$.

Clustering is initiated based on a random selection of k centroids. We assign each $x_{i,j}$ object to a particular cluster based on the cluster membership value w.r.t. a proximity measure, i.e., $dist(x, y)$. We use Euclidean distance as proximity measure. $dist$ is defined as

$$dist(x, y) = \begin{cases} 0 & if \text{ x = y} \\ \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2} & otherwise. \end{cases} \tag{5.5}$$

5.5.4 AOCD: The Algorithm

The AOCD algorithm is based on the NADO [5] approach. AOCD differs from NADO technique in the following key points

- We use the Principal Components Analysis (PCA) [326] feature reduction

technique to identify the relevant feature set. These feature sets are used during cluster formation (Algorithm 2).

- AOCD uses a variant of the *Fuzzy C-means* clustering algorithm to form clusters.
- We test AOCD using real-life coordinated datasets.
- AOCD adaptively updates the profiles for new test instances.

AOCD works as follows. $\{C_{11}, C_{12}, C_{13}, \cdots C_{1k}, C_{21}, C_{22}, C_{23}, \cdots C_{2k}, \cdots C_{s1}, C_{s2}, C_{s3}, \cdots C_{sk}\}$ is the set of clusters with cardinality sk. The method generates the profiles, $\mu_{s1}, \mu_{s2}, \mu_{s3}, \cdots \mu_{sk}$ for the clusters $C_{s1}, C_{s2}, C_{s3}, \cdots C_{sk}$, respectively, obtained from the dataset x. Then it detects coordinated scans based on the outlier score ROS' from the testing datasets. The major steps of AOCD are given in Algorithm 3.

Algorithm 3 AOCD (x, δ_2)

Input: x is the dataset, δ_2 is the threshold
Output: $O_{i,j}$'s are the anomalous objects

1: Select random sample, $x_1, x_2, \cdots, x_s$ from the dataset x using α''.
2: Find clusters $C_{s1}, C_{s2}, C_{s3}, \cdots C_{sk}$ for each sample x_s based on a variant of *Fuzzy C-means* clustering (Algorithm 2) technique w.r.t. relevant feature set F'.
3: Compute range based profile μ_{sk} for each of the sk clusters w.r.t. F'.
4: Calculate outlier score ROS' for each candidate data object, $X_{t_{i,j}}$ w.r.t. F' and μ_{sk}.
5: Rank the candidate data objects according to their score values.
6: Sort the data objects based on score values and report the anomalies or outliers, $O_{i,j}$'s w.r.t. the threshold δ_2.
7: **if** new test instances found **then**
8: Update range based profiles, μ_{sk}.
9: Return to Step 4.
10: **end if**

5.5.5 Complexity Analysis

AOCD has works in two basic modules: clustering and score computation. The clustering module takes O(kI) time to generate k clusters within I iterations. On the other hand, the score computation module takes $O(nklogk)$ time to estimate

the score value of each candidate data objects, where k is the number of reference points. Hence, the total time complexity of AOCD is $O(kI + nklogk)$.

5.6 Experimental Results

The main goal of the experiments is to apply AOCD to coordinated scan detection as well as to evaluate its capability in detecting outliers or anomalies or scans and compare it to the current best performing algorithms. To achieve this goal, we have implemented our algorithm and tested it with various real world datasets and datasets prepared by us on our TUIDS testbed at both packet and flow levels. It has been used during attack generation in our TUIDS testbed for labeled coordinated dataset preparation. The network laboratory layout where we capture network traffic for coordinated port scans data is shown in Figure 5.9. The network has 32 subnets including a wireless network, 4 routers, 3 wireless controllers, 8 L3 switches, 15 L2 switches and 300 hosts. The DHCP server is set up inside the main network for wireless network. During attack generation, we use 10 attackers per 32 subnets including one wireless subnet to launch the attacks.

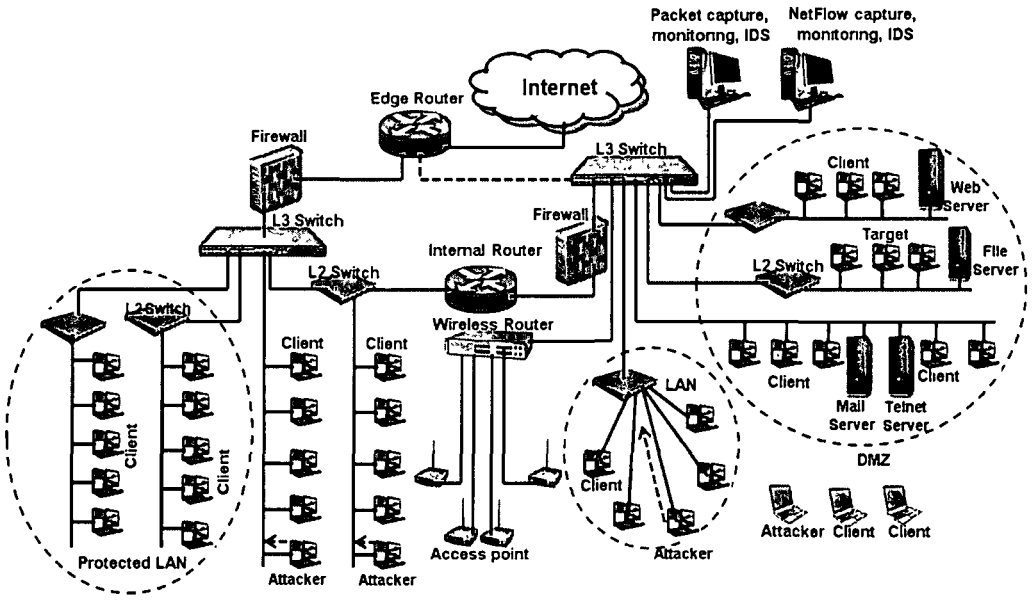


Figure 5.9: Coordinated port scan TUIDS testbed setup with layered 10 attackers /32 subnet including one wireless subnet

AOCD is implemented on an HP $xw6600$ workstation, Intel Xeon Processor

(3.00 GHz) with 4GB RAM. *Java* 1.6.0 is used for the implementation on Ubuntu 10.10 (Linux) platform. Java is used to facilitate the visualization and reusability of code for further experimentation.

5.6.1 Datasets Used

To evaluate the performance of AOCD, we use several real-life datasets for experimentation. We use three datasets: our own datasets that are *packet* and *flow* based and *KDDcup99 probe* [52] dataset. The characteristics of our own packet and flow level coordinated port scan datasets are presented in Table 5.4. The characteristics of the KDDcup99 probe datasets used in this experiments are given in Table 5.5.

Table 5.4: Distribution of normal and attack connection instances in TUIDS real-life packet and flow level intrusion datasets

	Dataset type			
Connection type	*Training dataset*		*Testing dataset*	
Packet level				
Normal	71785	100%	47895	75.78%
Probe			15307	24.22%
Total	71785	-	63202	-
Flow level				
Normal	23120	100%	16770	48.56%
Probe			17762	51.44%
Total	23120	-	34532	-

Table 5.5: Distribution of normal and attack connection instances in KDDcup99 probe datasets

	Dataset type			
Connection type	*Training dataset* (10% corrected)		*Testing dataset* (Corrected)	
Normal.	97278	100%	60593	87.98%
Probe.			8273	12.01%
Total.	97278	-	68866	-

5.6.2 Results and Discussion

We use our feature datasets for experiments at both packet and flow levels. The datasets are generated in our network security laboratory as discussed earlier in chapter 4. At both packet and flow levels, we extract basic features, content based features, time based features and window based features discussed in Chapter 4. We

convert all categorical attributes into numeric form and then compute the $log_z(a_{i,j})$ of the largest values to normalize data objects, where z depends on the attribute values and $a_{i,j}$ represents the largest attribute values.

We have generated sixteen types of attacks (see Table 5.1) for coordinated scans. However, in this experiment we consider only four types of scans (viz., TCP SYN, window, XMAS, and NULL) in coordinated mode during testing with both packet and flow level datasets. The PCAF module selects the relevance feature subsets for both packet and flow level datasets (see in Table 5.6). PCAF reduces the dataset in dimension based on feature relevance Hence, the feature IDs are seen in sequence in Table 5.6. This reduced dataset is used by the cluster formation and coordinated scan detection modules. AOCD is evaluated in terms of accuracy and false positive rate (FPR). The evaluation metrics are described in Chapter 4. We report the detection accuracy and false positive rate as follows.

- $Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$
- $False\ Positive\ Rate\ (FPR) = \frac{FP}{FP+TN}$

Details of performance of AOCD for the real-life TUIDS packet and flow level coordinated scan datasets are given in Table 5.7 and shown in Figure 5.10. Our results are better than the results in Singh and Chun [317]. They had obtained greater than 90% accuracy using their method The performance of the AOCD algorithm is excellent in case of probe class for both packet and flow dataset We see in Table 5.7 that the average accuracy for SYN, window, XMAS and NULL classes in packet level is 99.02% and in flow level it is 98.50%. In addition, we tested AOCD on four coordinated scan datasets, but the Singh and Chun [317] method was tested only on TCP SYN scan.

Table 5.6: TUIDS packet and flow level intrusion datasets - selected feature set

Method	*#Features*	*Selected features*
Packet level		
PCAF	19	1. 2. 3. 4. 5. 6. 7, 8, 9, 10, 11, 12, 13, 14. 15, 16. 17, 18. 19
Flow level		
PCAF	24	1. 2. 3. 4. 5. 6. 7, 8, 9, 10, 11, 12, 13, 14, 15, 16. 17. 18. 19. 20. 21. 22. 23, 24

Table 5.7: The performance of AOCD on the packet and flow level TUIDS intrusion datasets

Type of traffic	*Correctly detected*	*Miss detected*	*Accuracy (%)*
Packet level			
Normal.	47257	638	98.61
Probe.	15158	149	99.02
Overall.	62415	787	98.75
Flow level			
Normal.	16358	412	98.16
Probe.	14496	266	98.50
Overall.	30854	678	97.85

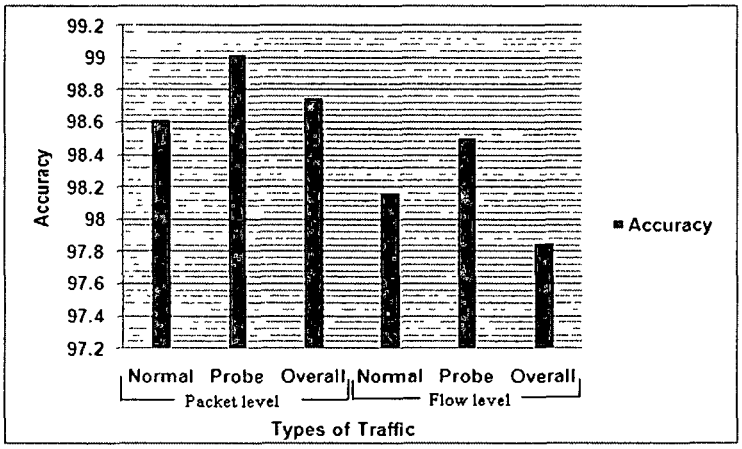


Figure 5.10: The performance of AOCD on the packet and flow level TUIDS intrusion datasets. The performance of flow level dataset is a bit lower than packet level dataset due to non-availability of packet specific information. But it is faster.

In another set of experiments, we use the KDDcup99 probe [52] dataset. Like the TUIDS datasets, we convert all categorical attributes to numeric and normalize them. We use KDDcup99 10% corrected normal dataset for training and KDDcup99 corrected and 10% corrected probe datasets for testing during performance analysis. The testing dataset contains six attacks, i.e., portsweep, ipsweep, satan, nmap, mscan and saint. The feature set selected by the PCAF module for normal and probe classes is given in Table 5.8. Here, we see a continuous sequence of feature IDs in Table 5.8 because of PCAF reduces the feature dimension. Performance details of these datasets are given in Table 5.9. Figure 5.11 shows the comparison of AOCD using the intrusion dataset with other similar algorithms, where the false positive rate is multiplied by 100 to highlight the efficiency of our approach in the graph. In our experiment, better results are obtained in KDDcup99 probe dataset with δ values in the range of (0.8 - 1.35) for normal records and (0.4 - 1.15) for

attack records.

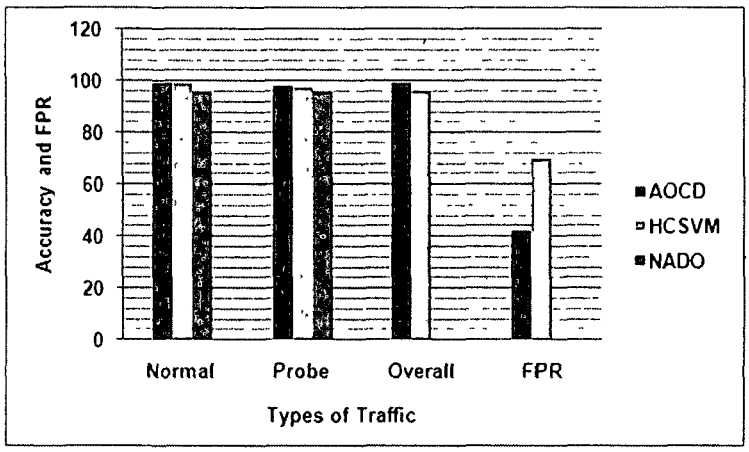


Figure 5.11: Comparison of the AOCD with other techniques using the KDDcup99 probe dataset. AOCD performs better than other two recent competing algorithms, HCSVM [4] and NADO [5] in terms of accuracy and false positive rates.

Table 5.8: KDDcup99 dataset - selected features set

Method	*#Features*	*Selected features*
Normal class		
PCAF	18	1, 2, 3, 4, 5, 6, 7, 8, 9. 10, 11, 12, 13, 14, 15. 16, 17, 18
FFSA [81]	6	5, 3, 1, 4, 34, 6
MMIFS [81]	6	5, 23, 3, 6, 35, 1
LCFS [81]	15	12, 34, 33, 3, 23, 27. 29, 40, 39, 28, 2, 41, 26, 35, 10
Probe class		
PCAF	25	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11. 12, 13, 14, 15. 16, 17, 18, 19. 20, 21, 22, 23, 24, 25
FFSA [81]	24	40, 5, 41, 11, 2, 22, 9, 27, 37. 28, 14, 19, 31, 18, 1, 17, 16. 13, 25, 39, 26, 6, 30. 32

Table 5.9: The performance of AOCD using the KDDcup99 probe dataset

Type of traffic	*Correctly detected*	*Miss detected*	*Accuracy (%)*
Normal.	60189	404	99.38
Probe.	8114	159	98.08
Overall.	68303	563	99.18

5.7 Summary

This chapter has examined the state-of-the-art of modern port scan techniques and approaches to detect them. The discussion follows well-known issues in scan

detection: detection strategies, data sources and data visualization. Experiments demonstrate that for different types of port scan attacks, different anomaly detection schemes may be successful Research prototypes combining data mining and threshold based analysis for scan detection have shown great promise. Such detection approaches tend to have lower false positive rates, scalability, and robustness.

In this chapter, we introduce an adaptive outlier based approach for coordinated port scan detection [5] is introduced. Unlike previous approaches which have been based on clustering and manual analysis, AOCD uses random sample selection using a linear congruential generator for distinct profile generation. In addition to this, we introduce an outlier score function to test each candidate object to identify coordinated port scan using score values This method classifies each candidate object as normal or coordinated port scan w.r t. a threshold. AOCD is capable of detecting coordinated scans that have a stealthy and horizontal or strobe footprint across a contiguous network address space We have tested this algorithm using several real-life datasets, viz., TUIDS datasets and KDDcup99 probe datasets. Coordinated scans are performed in an isolated environment, combining the network traffic traces with those collected from live networks. We extract various features from network packet as well as flow traffic data by developing our own modules for feature extraction. This approach achieves high detection accuracy and low false positive rates on various real-life datasets compared to existing coordinated scan detection approaches.

Chapter 6

Clustering and Outlier-based Approach for Network Anomaly Detection

This chapter starts by motivating the development of an clustering and outlier-based network anomaly detection method to detect anomalies with a high detection and few false alarm rate. To support outlier identification process, we introduce a tree-based clustering algorithm to generate a set of reference points. We use our outlier score function to rank each candidate object with respect to the reference points. Our technique selects relevant features from high dimensional datasets for use during cluster formation as well as during calculation of outlier scores for network anomaly identification. It reports the results as normal or anomalous with respect to an user defined threshold. We evaluate our approach in terms of detection rate, false positive rate, precision, recall, and F-measure using several high dimensional synthetic and real world datasets and find the performance to be effective in comparison to competing algorithms.

6.1 Introduction

There is growing need for efficient algorithms to detect exceptional patterns or trends and anomalies in network traffic data. The task of outlier discovery has five subtasks: (i) dependency detection, (ii) class identification, (iii) class validation, (iv) frequency detection, and (v) outlier or exception detection [329]. The first four

subtasks consist of finding patterns in large datasets and validating the patterns Techniques for association rules classification, and data clustering are used in the first four subtasks The fifth subtask of outlier detection focuses on a very small percentage of data objects, which are often ignored or discarded as noise Outlier detection techniques focus on discovering infrequent pattern(s) in the data. as opposed to many traditional data mining techniques such as association analysis or frequent itemset mining that attempt to find patterns that occur frequently

Outliers may represent aberrant data that may affect systems adversely such as by producing incorrect results, incorrect models, and biased estimation of parameters Outlier detection enables one to identify them prior to modeling and analysis [330] There are many significant applications of outlier detection For example, in the case of credit card usage monitoring or mobile phone monitoring a sudden change in usage pattern may indicate fraudulent usage such as stolen cards or stolen phone airtime Outlier detection can also help to discover critical entities For example, military surveillance, the presence of an unusual region in a satellite image in an enemy area could indicate enemy troop movement Most outlier detection algorithms make the assumption that normal instances are far more frequent than outliers or anomalies Generally network intrusion detection techniques are of two types *signature based* and *anomaly based* *Signature based* detection aims to detect intrusions or attacks from known intrusive patterns On the other hand, *anomaly based* detection looks for attacks based on deviations from established profiles or signatures of normal activities Events or records that exceed certain threshold scores are reported as anomalies or attacks Signature based intrusion detection cannot detect new or unknown attacks On the other hand, anomaly based detection techniques detect unknown attacks based on the assumption that the attack data deviate from normal data behavior However a drawback of anomaly based systems is high false alarm rates Minimization of the percentage of false alarms is the main challenge in anomaly based network intrusion detection. An outlier detection technique is effective in reducing the false positive rate with a desirable and correct detection rate

6.1.1 Motivation and Contributions

There are many outlier detection techniques [1, 105, 111, 329, 331, 332] in the literature, developed based on distance, density or combination of both, as well as soft computing and statistical measures Only some techniques have been applied to network anomaly detection and their detection rate is poor. In addition, even if a general outlier detection technique is tuned for network intrusion detection, it cannot perform well in high dimensional large datasets, because the classes are embedded inside subspaces. In case of score based outlier detection techniques, the score values do not vary w r.t. changes in the candidate objects during testing. So, it becomes very difficult to assign a label as normal or outliers. To address all these issues, we develop an efficient outlier based technique to analyze high dimensional and large amount of network traffic data for anomaly detection. Our technique has several features such as following (i) It is flexible enough to use any proximity measure during clustering and outlier score computation. (ii) It uses a subset of features to reduce the computation overhead (iii) our outlier score function can identify the network anomalies or outliers with low false alarm rate in addition to excellent identification of general outliers. (iv) The proposed technique can distinguish closely spaced objects during testing in terms of their score values. (v) It performs well for all types attacks when applied to network intrusion datasets. Specifically, the technical contributions of this chapter include the following.

- Challenges in clustering high dimensional large network traffic data include handling of mixed type data and arranging the data in computationally efficient structures for analysis. For example, *protocol* is categorial and *byte count* is numeric. Another key issue is how to represent a distance function that incorporates subspaces to find meaningful clusters.

  We propose TreeCLUS, an effective tree-based clustering algorithm based on relevant subspaces computation, to form compact and overlapped clusters. It uses an information gain based technique [42] for finding a relevant subset of features in order to identify the respective classes accurately.

- We use our outlier score function and use it extensively to detect anomalies or outliers efficiently in real-life intrusion datasets. It exploits TreeCLUS

algorithm for generating reference points for each cluster. We apply this function in network traffic anomaly detection and find excellent results using several real-life network traffic datasets

- We evaluate our technique using several high dimensional datasets. These include (a) synthetic dataset, (b) several UCI ML repository datasets, (c) real-life TUIDS intrusion datasets at both packet and flow levels, (d) real-life TUIDS coordinated scan dataset at both packet and flow levels, (e) KDD-cup99 intrusion datasets, and (f) NSL-KDD intrusion datasets. The performance of the proposed technique is excellent in comparison with the existing techniques.

6.2 Prior Research

Although many anomaly detection techniques have been developed and evaluated [333, 334] during the last several years, reducing the false alarm rate is still a challenging task. Several supervised anomaly detection techniques are available. These include ADAM (Audit Data Analysis and Mining) [335], neural networks [336], and SVMs[4] but their detection rates do not meet requirements of real-time application. A triangle area nearest neighbors (TANN) based intrusion detection technique is proposed by [8]. They combine supervised and unsupervised learning techniques to detect attacks Another unsupervised technique known as CBUID is proposed by [7]. There are many outlier detection techniques and there is also substantial discussion of applications to network anomaly detection in the literature. We broadly classify outlier detection techniques into four types: (a) statistical, (b) distance based, (c) density-based and (d) soft computing based.

6.2.1 Statistical Techniques

There are a large number of techniques in the literature [105, 111] for statistics based outlier detection. In statistical techniques, data instances are modeled based on an assumption of the stochastic distribution and instances are determined as outliers depending on how well they fit the model. However, most researchers point out that

a key drawback of these types of algorithms is the use of univariate distributions for many real applications without knowing the underlying distribution at all

Wang *et al.* [337] propose an outlier detection technique for probabilistic data streams. They present a dynamic programming algorithm based pruning technique to detect outliers efficiently The authors experimentally establish that the technique is efficient and scalable to detect outliers on large probabilistic data streams. Barbara et al. [338] used a pseudo-Bayes estimator to enhance detection of novel attacks. The main advantage of the pseudo-Bayes estimator is that no knowledge about new attacks is needed since the estimated prior and posterior probabilities of new attacks are derived from normal and known attack instances. A statistical signal processing technique based on abrupt change detection is proposed in [18] to detect anomalies in network traffic. It provides an outlier detection algorithm for detecting anomalous patterns.

The advantages of statistical techniques include the following. (i) If the models are appropriately defined, high precision can be attained for outlier detection, (ii) They are scalable in terms of both dimensionality and the number of instances. iii) Statistical techniques can operate in an unsupervised setting without any need for labeled training datasets.

The disadvantages of statistical techniques are the following (i) Appropriate thresholding for deviation detection irrespective of the application domain is a difficult task. (ii) They may not work efficiently in case of skewed distributions of high dimensional data due to lack of sensitivity.

6.2.2 Distance-based Techniques

A distance-based outlier detection technique is presented by Knorr et al. [339]. They define a point to be a distance outlier if at least a user-defined fraction of the points in the dataset are farther away than some user-defined minimum distance from that point. In their experiments, they primarily focus on datasets containing only continuous attributes. Distance-based methods also apply clustering techniques to the whole dataset to obtain a specified number of clusters. Points that do not cluster well are labeled as outliers.

Angiulli et al [340] report a technique for detecting the top outliers in an unlabeled dataset and produce a subset of it known as the outlier detection solving set This solving set is used to predict the outlierness of new unseen objects The solving set includes a sufficient number of points that permit the detection of the top outliers by considering only a subset of all pairwise distances in the dataset Ertoz et al [53] develop another intrusion detection system known as MINDS using unsupervised anomaly detection techniques and supervised pattern analysis techniques to detect attacks from real network traffic ADAM [335] is a well known on-line network based IDS It can detect known as well as unknown attacks It builds the profile of normal behavior from attack-free training data and represents the profile as a set of association rules It detects suspicious connections according to the profile

Szeto and Hung [331] propose two algorithms (i) Randomization with faster cutoff update and (ii) Randomization with space utilization after pruning to reduce the running time of ORCA The techniques are tested with large and high dimensional real datasets They claim that their techniques are as fast as 1 4-2 3 times ORCA and also claim that the techniques are parameter insensitive Jiang et al [341] propose a hybrid technique that combines boundary-based and distance-based methods They define outliers in terms of rough set theory and develop an outlier detection algorithm The authors obtain satisfactory result with two real life datasets

The advantages of distance based techniques are (i) They are easy for practical implementation, (ii) They obtain better results in case of uniformly dense datasets, and (iii) They are scalable to larger datasets

The disadvantages of distance based techniques are (i) Obtaining independent distance thresholds for specific applications is difficult, and (ii) The selection of an appropriate proximity measure is a problem in outlier detection techniques

6.2.3 Density-based Techniques

Density-based techniques are capable of handling outlier detection in large volume data [1] In one such technique a local outlier factor (LOF) is computed for each

point The LOF of a point is based on the ratio of the local density of the area around the point and the local densities of its neighbors. The size of the neighborhood of a point is determined by the area containing a user-supplied parameter called minimum number of points ($MinPts$).

Koufakou and Georgiopoulos [342] describe a fast, distributed outlier detection strategy intended for datasets containing mixed attributes. The method takes into account the sparseness of the dataset. It is highly scalable with the number of points and the number of attributes in the dataset. A density-based technique takes into account the distribution of the input space. Outliers can be identified by considering the lower density regions in the neighborhood of each data instance. An instance that lies in a neighborhood with low density is declared an outlier while an instance that lies in a dense neighborhood is declared to be normal.

Density-based outlier detection techniques have also been used to detect anomaly behavior One such technique presented in [54] uses decision trees to develop a prediction model over normal data to detect anomaly. It exploits data mining techniques to discover consistent and useful patterns of system features that describe program and user behavior. It can recognize anomalies and known intrusions satisfactorily. A novel local distribution based outlier detection technique is proposed by Zhang et al. [343]. They report two algorithms LDBOD and LDBOD+ with three measures, viz., local-average-distance, local-density, and local-asymmetry-degree. Their algorithms seem to be better than LOF when used with intrusion datasets.

The advantages of density based techniques are: (i) They can be used to model the real world situation more realistically, and (ii) They are scalable in terms of dimensionality and number of instances.

The disadvantages of density based techniques are: (i) Outlier detection is highly sensitive to input parameters and it is difficult to estimate these parameters properly for all application domains, and (ii) Most techniques are insensitive to a dataset with variable density, i.e.. non-uniform density dataset.

6.2.4 Soft Computing Techniques

Soft computing techniques are being widely used by the IDS community due to their generalization capabilities that help in detecting known and unknown intrusions or attacks that have no previously described patterns. Researchers have used rule-based techniques for intrusion detection earlier, but had difficulty in detecting new attacks or attacks that had no previously described patterns [336] Therefore, the idea of using rough sets may be promising. The basic idea behind rough sets is the following: For a subset z of the universe and any equivalence relationship on the universe, the difference between the upper and lower approximations of z constitutes the boundary region of the rough set, whose elements cannot be characterized with certainty as belonging to z or not, using the available information. The information about objects from the boundary region is, therefore, inconsistent or ambiguous Based on available information, if an object in z always lies in the boundary region with respect to every equivalence relation, we may consider this object as not behaving normally according to the given knowledge at hand. Such objects are called *outliers* [115]. An outlier in z is an element that cannot be characterized with certainty as always belonging to z or not, using the given knowledge. Rough uncertainty is formulated in terms of rough sets [344].

Fuzzy and rough sets represent different facets of uncertainty Fuzziness deals with vagueness among overlapping sets. Rough sets deal with non-overlapping concepts; they assert that each element may have different membership values for the same class [344]. Thus, roughness appears due to indiscernibility in the input pattern set, and fuzziness is generated due to the vagueness present in the output class and the clusters. To model this type of situation fuzzy-rough sets are used. Xue et al. [332] introduce a semi-supervised outlier detection method using fuzzy-rough C-means clustering. The authors present an objective function, which minimizes the sum of squared errors in clustering, the deviation from known labeled examples as well as the number of outliers.

The advantages of soft computing based techniques are: (i) Such techniques are very effective and practical for classification, and (ii) They allow one to incorporate domain information effectively.

The disadvantages of soft computing based techniques are: (i) Non-parametric techniques are difficult to apply in outlier detection and (ii) The computational complexity is very high as the number of instances rises.

6.2.5 Discussion

Outlier detection has largely focused on univariate data, and data with a known distribution. These two main limitations have restricted the ability to apply outlier detection methods to large real world databases which typically have many different fields. We make the following observations.

- Most outlier detection techniques are statistical and their detection rates are very low.
- Most existing algorithms perform poorly in high dimensional spaces because classes of objects often exist in specific subspaces of the original feature space. In such situations, subspace clustering is expected to perform better.
- As the dataset size increases, the performance of existing techniques often degrades.
- Most existing techniques are for numeric datasets only and only a few algorithms are capable of detecting anomalies from mixed type data.
- The existing outlier detection techniques are often unsuitable for real time use.

A comparison of several existing outlier detection techniques is given in Table 6.1. Note that in comparing these works, we consider seven different classes of outliers the methods can detect as shown in Figure 6.1. These seven classes were introduced in [345] by us and they are defined formally in Subsection 6.5.4.

6.3 Problem Formulation

The problem is to analyze any application domain over an optimal and relevant feature space in order to identify all possible nonconforming patterns (if exist)

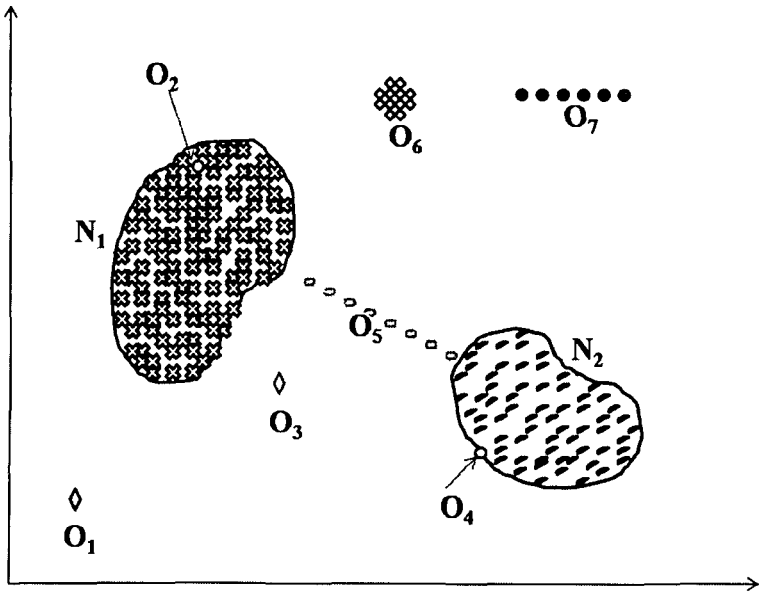


Figure 6.1: Illustration of seven different cases: N_1 and N_2 are two normal clusters. O_1 is the distinct outlier, O_2, the distinct inlier, O_3, the equidistance outlier, O_4. the border inlier, O_5, a chain of outliers, O_6 is another set of outlier objects with higher compactness among the objects and O_7 is an outlier case of "*stay together*".

Table 6.1: A generic comparison of existing outlier detection techniques

Method	*Year*	*U/M*	*Data Type*	*Outlier Case(s) handled*	*Technique*
DB Outliers [329]	2000	U	numeric	C1 C2, C4 C5	distance based
LOF [1]	2000	U	numeric	C1 C2 C4	density based
ODP & OPP [340]	2006	M	numeric	C1 C2 C4 C5	distance based
Rough Set [344]	2009	U	-	C1 C2 C4	soft computing
LDBOD [343]	2008	M	numeric	C1 C2 C5	density based
LSOF [346]	2009	M	numeric	C1 C2 C4	density based
ODMAD [342]	2010	M	mixed	C1 C2 C3, C4	density based
FRSSOD [332]	2010	M	numeric	C1 C2 C4 C5	soft computing
DIODE [347]	2010	M	numeric	C1 C2 C3 C4	distance based
RC & RS [331]	2010	M	numeric	C1 C2 C4	distance based
DPA & PBA [337]	2010	U	numeric	C1 C2 C4	statistics based
NADO [345]	2011	M	numeric	C6	distance based
BD Outliers [341]	2011	M	numeric	C1 C2 C3 C5	distance based

U/M Univariate (U)/Multivariate (M)
Outlier case(s) C1 - distinct outlier, C2 - equidistance outlier, C3 - chain outliers C4 - group outliers, C5 - stay together C6 - all outlier cases are discussed in Section 6 2

among the real-life instances with reference to a given normal behavior It assumes an instance x_i to be an outlier or nonconforming iff (a) $x_i \in C_i$ and $|C_i| \ll |C_i^N|$, where C_i represents the i^{th} group or set of outlier instances and C_i^N represents the i^{th} group or set of normal instances, $|C_i|$ represents the cardinality of the group C_i and similarly, $|C_i^N|$ represents the cardinality of the group C_i^N, and (b) the outlier score of $x_i > \tau$, where τ is a user-defined threshold

6.4 Theoretical Foundations

An outlier or anomaly is an abnormal or infrequent event or point that varies from the normal event or point. These patterns are often known as anomalies, outliers, exceptions, surprises, or peculiarities in different application domains Anomalies and outliers are two terms used most commonly in the context of anomaly detection, sometimes interchangeably. The importance of anomaly detection is due to the fact that anomalies in data translate to significant, and often critical, actionable information in a wide variety of application domains. For example, an anomalous traffic pattern in a computer network could mean that a hacked computer is sending out sensitive data to an unauthorized destination. A network administrator needs to define an abnormal event based on normal network statistics when he/she wants to detect network anomalies.

Outliers may occur due to several reasons including malicious activity. Unlike noise in data, outliers are interesting to the analyst due to their domain specific importance. In outlier based network anomaly detection, outliers are assumed to be anomalous instances. Outlier detection is an important technique in detecting exceptional events from small or large datasets In many data analysis tasks, outlier detection is one of the techniques used for initial analysis of the data. Even though outliers may be errors or noises, they may still carry important information in particular domains. For example, consider network traffic analysis. The importance of outlier detection is due to the following: (i) An outlier may indicate bad data but it may still be important to detect. (ii) An outlier may make data inconsistent, but again it is important to detect such cases.

When using outlier detection methods, it is necessary to compute an outlier score for each object to determine its degree of outlierness. An outlier score is a summarized value based on the distance, density or other statistical measures. There are different approaches for computing this outlier score.

Given a dataset, it is possible to compute outliers directly by computing the outlierness score for each point in the dataset. We can then identify points whose outlierness is above a certain user-provided threshold as outliers. However, finding outliers directly on a dataset has drawbacks For example, if a dataset has a large

number of features, it is not efficient to perform outlier score computation using all the features of the data items. Not only that, the outliers found may not be the best if all features are used because of useless features or interactions among features. Thus, the usual approach is to obtain a subset of features that are relevant to the problem at hand. When one computes outlier scores directly, the entire dataset is considered one class or group. Outlier finding becomes more efficient and also produces better results, if instead of considering the entire dataset as one class, we cluster the dataset first. Some of the clusters are likely to be bigger than others in terms of the number of items in them. For each cluster, big or small, we compute what is called its reference profile. We compute outliers with respect to these reference profiles Those data points whose outlier scores are above a certain user-given threshold are identified as real outliers or anomalous data points. To keep our discussion rigorous, we introduce some definitions and lemmas to explain our approach Let us first start by characterizing our dataset.

Definition 6.4.1. Dataset: *A dataset* $\mathbb{X}$ *is a set of* n *objects, i.e.,* $\{x_1, x_2, x_3 \cdots x_n\}$, *where each object* x_i *is represented by a* d*-dimensional vector, i.e.,* $\{x_{i,1}, x_{i,2}, x_{i,3} \cdots x_{i,d}\}$, *where* x_{ij} *can be a numeric or categorical attribute.*

Before any processing, we compute an optimal subset of features, as discussed later in the chapter. We denote the dataset described by the reduced set of features $\mathbb{X}$ as well. As indicated earlier, clustering is a step in our approach. To cluster the dataset $\mathbb{X}$, we need to use a distance measure, which is used to compute similarity between a pair of data objects. Our method works with any distance measure *dist*. We use a threshold α' for computation of similarity.

Definition 6.4.2. Object Similarity: *Two data objects* x_1 *and* x_2 *are defined as similar iff (a)* $dist(x_1, x_2) \ll \alpha'$ *and (b)* $dist(x_1, x_2) = 0$, *if* $x_1 = x_2$.

The notion of object similarity is necessary to cluster the dataset $\mathbb{X}$.

Definition 6.4.3. Cluster: *A cluster* C_i *is a subset of a dataset* $\mathbb{X}$, *where for any pair of* x_i*'s, say* $(x_i, x_j) \in C$, *dist* $(x_i, x_j) \ll \alpha'$.

Once the dataset has been clustered, we create a cluster profile for each of our clusters. These profiles are also called reference profiles.

Definition 6.4.4. Profile: *The profile of a cluster C_i is a mean representation of i^{th} cluster, $\mu_i^1, \mu_i^2, \cdots, \mu_i^m$ over an optimal subset of features with cardinality m.*

Before we proceed to discover outliers in a dataset, it behooves us to characterize an outlier and the types of outliers we are interested in, so that our approach is as comprehensive as possible. We are focussed on seven different types of outliers [345] shown in Figure 6 1 for the evaluation of our method for outlier detection. Outliers are computed by computing an outlier score for each of the objects under question. To overcome the drawbacks of existing outlier score function such as *ROS* [3], we use our enhanced outlier score function called *ROS'* presented later in Subsection 6.5 4.

6.5 The Proposed Technique

The proposed technique aims to detect anomalous patterns from real life network traffic data with high detection rate It works by identifying reference points and by ranking based on outlier scores of candidate objects. The proposed technique has four parts including: feature selection, clustering, profile generation and outlier detection. Figure 6.2 shows the framework for out outlier based technique to network anomaly detection We describe each part of the algorithm in detail in the subsections that follow

6.5.1 Feature Selection

Feature selection is an essential component of classification based knowledge discovery. Feature selection is a multi-step process that aims to identify an optimal subset of features say, $F' \subseteq \mathbb{F}$, i.e., the set of all features for a dataset $\mathbb{X}$, which gives best possible accuracy on $\mathbb{X}$. Identification and use of relevant features for the input data leads to reduction in storage requirement, reduction of computational cost, simplification of the problem, and increase in accuracy. There are two main models for feature selection: filter methods and wrapper methods [42, 76, 81, 348, 349]. The filter methods are usually less expensive than wrapper methods and are also able to scale up to large datasets We use an information gain based method [42] for the

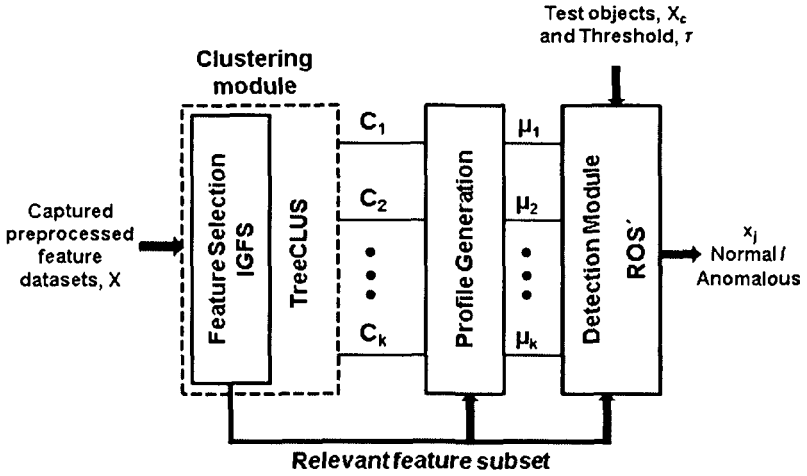


Figure 6.2: A framework for outlier-based network anomaly detection. The framework takes feature dataset $\mathbb{X}$ as input. It initially selects relevant and optimal feature subset by IGFS. TreeCLUS uses these feature subset during cluster formation, $C_1, C_2, \cdots, C_k$. Once cluster formation is over, it generates mean-based profiles, $\mu_1, \mu_2, \cdots, \mu_k$ from each cluster. Finally, ROS′ computes the outlier score for each test object, x_c and report as normal or anomalous based on a user defined threshold, τ.

identification of relevant features prior to clustering, profile generation and outlier detection

We have experimented with several feature selection methods and give a performance comparison of the methods available with Weka[1], viz., correlation based feature selection (CFS), principal components analysis based feature selection (PCAFS) and information gain based feature selection (IGFS) using the KDDcup99 intrusion dataset in Figure 6.3 As seen from the figure, As seen from the figure, although a sharp variation exist in terms of detection rate for different classes of attacks as well as normal, the IGFS (shown with solid line) is always performs better including normal class identification in comparison to CFS and PCAFS. Also, IGFS identifies minimum relevant feature subset for all classes, viz , normal (10-features), DoS (12-features), probe (14-features), R2L (13-features), and U2R (15-features) than CFS and PCAFS. Hence, we select the information gain based method [42] for relevant and optimal feature selection Suppose S is a set of training samples with corresponding class labels. Let there be m classes. The training set contains s_i samples of class I and s is the total number of samples in the training set. The amount of information contained in the training set is quantified using the following

[1]http //www cs waikato.ac nz/ml/weka/

formula [42]

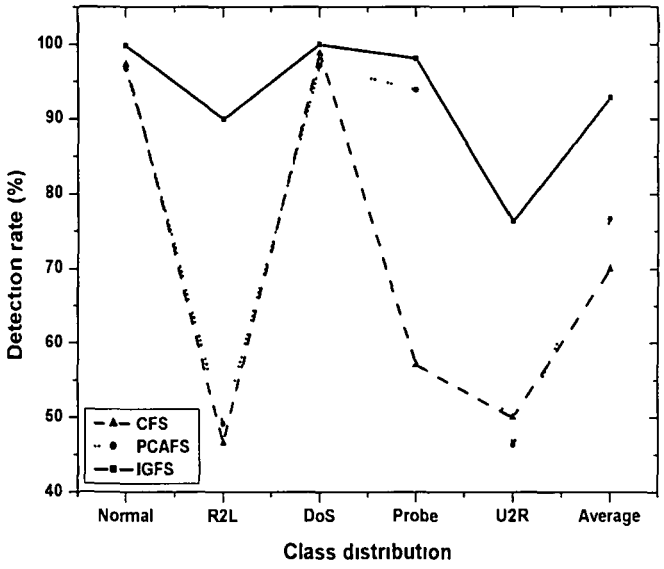


Figure 6.3: A comparative performance analysis of correlation based PCA based and IGFS based relevant feature selection techniques using KDDcup99 intrusion dataset

$$I(s_1\ s_2,\quad s_m) = -\sum_{i=1}^{m} \frac{s_i}{s} log_2(\frac{s_i}{s}) \tag{6 1}$$

Let $\mathbb{F} = \{f_1\ f_2\quad f_d\}$ be the set of features in the samples in the training set Thus, the training set is described in terms of d features Not every feature appears in every sample in the training set Let S_j be the subset of training samples that contain feature f_j In addition, let S_j contain s_{ij} samples of class i Then, the entropy of the feature $\mathbb{F}$ is defined [42] as

$$E(\mathbb{F}) = \sum_{j=1}^{v} \frac{s_{1j} + s_{2j} + \quad + s_{mj}}{s} \times I(s_{1j}, s_{2j}\quad, s_{mj}) \tag{6 2}$$

Information gain can be calculated as

$$Gain(\mathbb{F}) = I(s_1\ s_2,\quad s_m) - E(\mathbb{F}) \tag{6 3}$$

The algorithm takes the labelled dataset $\mathbb{X}$ and threshold, ξ as input It returns the selected features, F' The major steps of this technique are given in

Algorithm 12.

Algorithm 4 : IGFS $(\mathbb{X}, \xi)$

Input: $\mathbb{X}$ is the labelled dataset and ξ is the threshold
Output: F' is the selected feature set

1: initialization. $\mathbb{F} \leftarrow$ set of all features, $\mathbb{C} \leftarrow$ set of all classes
2: **for** each feature $f_i \in \mathbb{F}$ **do**
3: compute $I(f_i, \mathbb{C})$ ▷ f_i is the i^{th} feature, and $\mathbb{C}$ is the class label
4: compute entropy of feature f_i, $E(\mathbb{F})$, where $\mathbb{F} = \{f_1, f_2, \cdots f_d\}$ ▷ $E(\mathbb{F})$ is the entropy of all features
5: **end for**
6. calculate information gain, $G(\mathbb{F})$ and the subset of features maximizes, $G(\mathbb{F})$ with respect to the threshold ξ
7 selected feature set F'

6.5.2 The Clustering Technique

After we have expressed our input data in terms of the selected relevant feature subset using Algorithm 12, we apply TreeCLUS, a tree based clustering technique to hierarchically arrange the dataset, $\mathbb{X}$ into k clusters, viz., $C_1, C_2, C_3, \cdots C_k$. The clusters formed are used to generate profiles prior to outlier detection in high dimensional network traffic datasets for anomaly detection. TreeCLUS mainly depends on two parameters α' and β', which are thresholds used for initial node formation It expands a node in depth-wise by reducing the subset of features based on the decreasing order of information gain value at feature space to get a specific class.

TreeCLUS generates a tree by creating nodes in a depth-first manner with all the objects as the root The root is at level 0 and is connected to all the nodes in level 1. Each node in level 1 is created based on a maximal relevant subspace with an arbitrary unclassified object, x_i, w.r.t a threshold α' for proximity measure and the neighborhood of x_i w r.t. a threshold β' to form a cluster. If no object is found to satisfy the neighborhood condition β' and proximity measure α' with x_i in reduced space, the process restarts with another unclassified object. The process continues till no more object can be found to assign in a node The major steps of the subspace based TreeCLUS clustering technique are given in Algorithm 5, 6.

For example, let $\mathbb{X}'$ be a sample dataset shown in Table 6.2. $O_1, O_2, \cdots O_{16}$ are

Algorithm 5 Part1 TreeCLUS ($\mathbb{X}, \alpha', \theta$)

Input: $\mathbb{X}$, dataset, α' threshold for node formation, θ, height of the tree
Output: generate clusters, C_1 C_2, C_3 C_k

```
1  initialization node_id ← 0                          ▷ node_id is increased by 1 for new node
2  function BUILDTREE(𝕏, node_id)                      ▷ function to build tree
3      for i ← 1 to x do
4          if (x_i classified ! = 1 and check_ini_feat(IGFS(x_i)) == true and dist ≤ α'
   and nb ≤ β') then ▷ check_ini_feat() is the function to check features set, nb is
   the neighborhood condition
5              CreateNode(r_i no, p_id, temp, node_count node_id, l)   ▷ function to
   create new node
6              while (n_F' - (l - 1)) ≥ θ do            ▷ check relevant features subset
7                  l++
8                  for i ← 1 to 𝕏 do
9                      if r_i classified ! = 1 then ▷ if object is classified then labelled
   as 1
10                         p_id = check_parent(x_i no, l) ▷ function to check parent id
11                         if (p_id > -1 and check_ini_feat(IGFS(x_i)) == true) then
12                             CreateNode(x_i no, p_id, temp, node_count, node_id, l)
13                         end if
14                     end if
15                 end for
16             end while
17             l = 1                                    ▷ initially node level is 1
18         end if
19     end for
20 end function
21 function CREATENODE(no, p_id, temp, node_count id, l)   ▷ function to create
   node
22     node_id = new node()
23     node_id temp = temp
24     node_id node_count = node_count                  ▷ number of nodes in a level
25     node_id p_node = p_id
26     node_id id = id,
27     node_id level = l                                ▷ set level l
28     ExpandNode(no, id node_id temp, node_count, l)   ▷ expand node in
   depth-wise for a particular node
29     temp = NULL
30     node_count = 0
31     node_id++
32 end function
33 function EXPANDNODE(no, id, temp, node_count, l) ▷ function to expand node
34     if r_no classified == 1 then
35         return
```

Algorithm 6 Part 2 TreeCLUS ($\mathbb{X}$ $\alpha', 0$)

```
36    else
37        x_no classified = 1
38        τ_no node_id = id
39        for ι ← 1 to τ do
40            if (x_ι classified ! = 1) then
41                minRank_F' = find_minRank(IGFS(x_ι))    ▷ select next subset of
    relevant features
42                if (n_F' - minRank_F') ≥ 0 then  ▷ check maximum height of the
    tree
43                    for j ← minRank_F' to n_F' do  ▷ continue for getting specific
    class
44                        ExpandNode(x_ι no, id, temp  temp_count, l)  ▷ expand node
    in depth-wise
45                    end for
46                end if
47            end if
48        end for
49    end if
50 end function
```

the objects f_1 f_2 f_{10} are the features and CL is the class label We find the relevant feature set from $\mathbb{X}'$ by using Algorithm 12 shown in Table 6 3 R is the root containing the total number of objects A tree (see Figure 6 4) is obtained from the sample dataset $\mathbb{X}'$ by using the TreeCLUS algorithm From Table 6 3 and Figure 6 4, we can see that in level 1 it creates three nodes with classes C_1 C_2 and C_3, and in level 2 it creates eight nodes with classes C_{11}, C_{12}, C_{21}, C_{22}, C_{23} C_{31}, C_{32} and C_{33} w r t the selected relevant feature subset Due to the smaller dataset, the variance in the rank values are low If the rank value is the same, we choose the first set of features for obtaining finer clusters otherwise, we choose higher rank values

The proposed clustering algorithm, TreeCLUS, differs from earlier work [345] in clustering and is also more suitable for our purpose

- Unlike some of the well-known partitioning-based clustering algorithms, like k-means [350] and its variants PAM, CLARA CLARANS, the proposed TreeCLUS doesn't require the number of clusters as input a priori However, like some effective hierarchical clustering algorithms, it accepts the height of the tree as input parameter for termination of cluster expansion

Table 6.2: Sample dataset, $\mathbb{X}'$

Object ID	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	*CL*
O_1	9 23	0 71	2 43	0 60	104	2 80	3 06	0 28	2 29	5 64	1
O_2	22 53	6 51	6 64	4 96	72 79	2 60	11 63	0 80	9 00	36 97	8
O_3	16 37	5 76	1 16	11 88	95	5 50	1 10	0 49	6 87	20 45	5
O_4	10 37	1 95	9 50	0 80	110	1 85	2 49	0 64	3 18	9 80	2
O_5	14 67	4 85	1 92	11 94	96	4 10	1 79	0 12	4 73	10 80	4
O_6	9 20	0 78	2 14	0 20	103	2 65	2 96	0 26	2 28	4 38	1
O_7	12 37	0 84	1 36	11 60	95	3 98	1 57	0 98	1 42	10 95	3
O_8	9 16	1 36	9 67	0 60	110	1 80	2 24	0 60	3 81	9 68	2
O_9	16 17	5 86	1 53	11 87	93	5 89	1 75	0 45	6 73	20 95	5
O_{10}	18 81	6 31	4 40	4	70	2 15	8 09	0 57	7 83	27 70	6
O_{11}	14 64	4 82	1 02	11 80	94	4 02	1 41	0 13	4 62	10 75	4
O_{12}	20 51	6 24	5 25	4 50	70 23	2	9 58	0 60	8 25	32 45	7
O_{13}	12 33	0 71	1 28	11 89	96	3 05	1 09	0 93	1 41	10 27	3
O_{14}	20 60	6 46	5 20	4 50	71	2 42	9 66	0 63	8 94	32 10	7
O_{15}	18 70	6 55	5 36	4 50	73 24	2 70	8 20	0 57	7 84	27 10	6
O_{16}	22 25	6 72	6 54	4 89	69 38	2 47	10 53	0 80	9 85	36 89	8

Table 6.3: Relevant feature set and attribute rank values

Class	*Object ID*	*Relevant feature set*	*Feature rank value*
C_1	O_1,O_4 O_6,O_8	f_5,f_6,f_2 f_3,f_9,f_{10},f_7 f_8	1,1,1,1,1,1,1,1
C_{11}	O_1,O_6	f_5,f_6,f_2 f_3,f_9,f_{10},f_7	1,1,1,1,1,1,1
C_{12}	O_4,O_8	f_5,f_6,f_2 f_3,f_9,f_{10}	1,1,1,1,1,1
C_2	O_3,O_5 O_7,O_9,O_{11},O_{13}	f_1, f_2 f_6, f_9, f_8, f_{10}	1 585,1 585,1 585,1 585 1 585,0 918
C_{21}	O_3, O_9	f_1, f_2 f_6, f_9, f_8	1 585,1 585,1 585,1 585 1 585
C_{22}	O_5, O_{11}	f_1, f_2 f_6, f_9	1 585,1 585,1 585,1 585
C_{23}	O_7, O_{13}	f_1, f_2 f_6, f_8	1 585,1 585,1 585,1 585
C_3	O_2,O_{10},O_{12},O_{14},O_{15} O_{16}	f_7, f_1 f_{10}, f_8, f_9, f_4, f_3	1 584,1 584,1 584,1 584 1 584,0 917,0 917
C_{31}	O_2, O_{16}	f_7, f_1 f_{10}, f_8, f_9, f_4	1 584,1 584,1 584,1 584 1 584,0 917
C_{32}	O_{10}, O_{15}	f_7, f_1 f_{10}, f_8, f_9	1 584,1 584,1 584,1 584 1 584
C_{33}	O_{12}, O_{14}	f_7, f_1 f_{10}, f_8, f_4	1 584,1 584,1 584,1 584 0 917

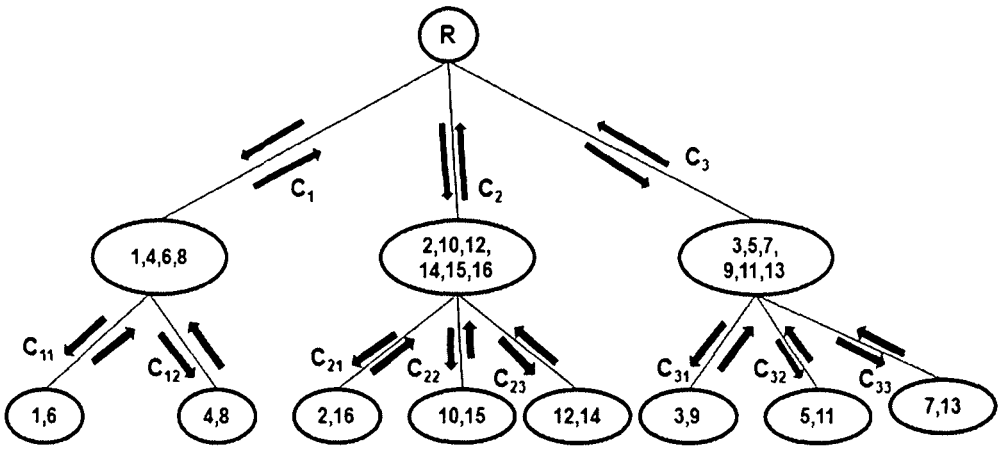


Figure 6.4: Tree generated from $\mathbb{X}'$ $\mathbb{C}$ represents class and a node contains the object IDs w r.t a class

- Unlike most density-based clustering algorithms [351], TreeCLUS is scalable both in terms of number of instances as well as dimensions

- Like those few subspace clustering algorithms [31], the proposed TreeCLUS

also works on subspaces to handle high dimensional numeric data

- It can also identify overlapped clusters in addition to disjoint clusters like [352] which is helpful in the identification of network traffic anomalies

- For empirical evaluation of our TreeCLUS with the other traditional clustering algorithms, all the algorithms are executed over full feature space The agglomerative hierarchical clustering algorithm [353] is implemented using average linkage distance measure while fuzzy c-means [354] is implemented using common parameter setting, i e , $m = 2$ as fuzziness coefficient $l = 69$ as number of iterations and $\varepsilon = 0\ 01$ as stopping criteria However, in case of [345] we implemented this algorithm using maximum correlation-based initial centroid selection technique [345], where we set $k = 25$ as the number of clusters, $l = 44$ as the number of iterations A comparison of these algorithms with TreeCLUS is shown in Figure 6 5 It can be observed from the figure that TreeCLUS works better than its other competing algorithms

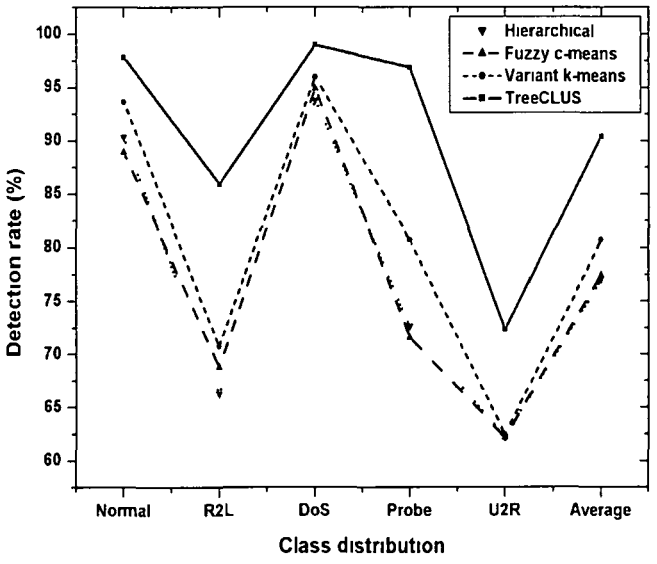


Figure 6.5: A performance comparison of TreeCLUS with similar clustering techniques using KDDcup99 intrusion dataset

6.5.3 Profile Generation

Once cluster formation is over, we calculate the reference points, R_k for each cluster ($i.e.$, C_1 C_2, C_3 C_k) We compute the mean of each feature value over the reduced feature space that corresponds to a cluster C_i (where $i = 1, 2\ 3, \cdots, k$) to compose the profile for C_i If μ_1^i is the i^{th} mean of the profile corresponds to the cluster C_1 then for an optimal feature space of cardinality m for cluster C_1, the profile representation will be $\{\mu_1^1, \mu_1^2\ \mu_1^3, \cdots, \mu_1^m\}$ The major steps in profile generation are given in Algorithm 7

Algorithm 7 ProfileGEN ($\mathbb{C}_k$)

Input: $\mathbb{C}_k$ is the clusters generated by TreeCLUS from dataset $\mathbb{X}$
Output: generate profiles $\mu_1, \mu_2, \mu_3,$ μ_k

```
1 select relevant feature set, F' from the dataset x by using IGFS()
2 call TreeCLUS(), k ← |ℂ|, where ℂ is the set of clusters        ▷ k is the total number of clusters
3 for i ← 1 to k do                                              ▷ for each k number of clusters
4     for j ← 1 to C_i do                                        ▷ for each cluster C_i
5         compute μ_ij(F') = Σ_{i=1}^{N_C} (x_jF' / N_C), where F' = f_1 f_2, · · f_d'   ▷ j is the attributes of each object within a class C_i
6     end for
7     build mean based profiles μ_i                              ▷ for each cluster C_i, profiles, μ_i
8 end for
```

6.5.4 Outlier Detection

The technique assumes a normality model [64] and considers larger clusters as normal and smaller clusters as outliers We assume that larger clusters $C_1, C_2, C_3, \cdots C_m$, say, are normal and smaller clusters (may include singleton clusters) are outliers or anomalies Let S_i be the number of classes to which each of k' nearest neighbor data objects belongs k' plays an important role in score computation Therefore, the selection of k' values for different datasets is discussed separately in Subsection 6 6 2 Let x_c be a data object in $\mathbb{X}_c$ and $dist(x_c, R_i)$ be the distance between the data object x_c and the reference points R_i, where $c = 1\ 2, 3, \cdots n$, $dist$ is a proximity measure used, and $\mathbb{X}_c$ represents the whole candidate dataset. The technique works well with any commonly used proximity measure The outlier score for a data object x_c we define is given in Equation(6 4)

$$ROS'(x_c) = \frac{\max_{1\le i\le k'} S_i}{k'} \times \left(\frac{\left(1 - \min_{1\le i\le k'} dist(x_c\ R_i)\right) \times \left(\sum_{i=1}^{k'} \min_{1\le i\le k'} dist(x_c, R_i)\right)}{\sum_{i=1}^{k'} \max_{1\le i\le k'} dist(x_c\ R_i)} \right) \quad (6\ 4)$$

Here $\frac{\max_{1\le i\le k'} S_i}{k'}$ is the maximum probability that a data object belongs to a particular class, the remaining part is the summarized value of the distance measure with k' nearest neighbors over the relevant feature subset The candidate data objects are ranked based on outlier score values Objects with scores higher than a user defined threshold τ are considered anomalous or outliers

Definition 6.5.1. Outlier Score - *An outlier score ROS' with respect to a reference point is defined as a summarized value that combines distance and maximum class occurrence with respect to k' nearest neighbors of each candidate data object (See formula in Equation*(6 4)*)*

Having defined an acceptable outlier score, we first define inliers or normal instances, and then outliers of different kinds For a visual interpretation of these definitions, the reader should refer to Figure 6 1

Definition 6.5.2. Distinct Inlierness - *An object O_i is defined as a distinct inlier if it conforms to normal objects, i e , $ROS'(O_i\ C_i) \ll \tau$ for all i*

Definition 6.5.3. Border Inlierness - *An object O_i is defined as border object in a class C_i, if $ROS'(O_i, C_i) < \tau$*

Definition 6.5.4. Outlier - *An object O_i is defined as an outlier w r t any normal class C_i and the corresponding profile R_i iff (i) $ROS'(O_i, C_i) \geq \tau$, and (ii) $dist(O_i, R_i) > \alpha'$, where α' is a proximity based threshold, and dist is proximity measure*

Definition 6.5.5. Distinct Outlierness - *An object O_i is defined as a distinct outlier iff it deviates exceptionally from the normal objects, i e , from the generic class C_i In other words, $ROS'(O_i, C_i) \gg \tau$ for all i*

Definition 6.5.6. Equidistance Outlierness - *An object O_i is defined as equidistance outlier from classes C_i and C_j, if $dist(O_i\ C_i) = dist(O_i, C_j)$ but $ROS'(O_i, C_i) > \tau$*

Definition 6.5.7. Chain Outlierness - *A set of objects, $O_i\ O_{i+1}, O_{i+2} \cdot$ is defined as a chain of outliers if $ROS'\ (O_{i+l}, C_i) \geq \tau$, where $l = 0, 1, 2, \cdot\ \cdot, z$*

To ensure that our approach for finding outliers works correctly, we prove two lemmas below.

Lemma 6.5.8. *If O_i and O_j are two outlier objects with distinct neighborhoods, then with reference to a given class C_i, $ROS'(O_i) \neq ROS'(O_j)$*

Proof Assume that O_i and O_j are two outlier objects with reference to a given class C_i and $ROS'(O_i, C_i) = ROS'(O_j, C_i)$. As per *Definition* 6 5 4, O_i and O_j are outliers w.r.t class C_i, iff $ROS'(O_i, C_i)$ or $ROS'(O_j, C_i) \geq \tau$ As per *Definition* 6.5.1, ROS' is estimated based on the Equation(6.4), where $\frac{\max_{1 \leq i \leq k'} S_i}{k'}$ plays an important role. The value of $\frac{\max_{1 \leq i \leq k'} S_i}{k'}$ for a candidate object will vary with different neighborhoods. Hence, $ROS'(O_i) \neq ROS'(O_j)$ w.r.t. class C_i. □

Lemma 6.5.9. A given outlier object O_i cannot belong to any of the normal clusters i.e., $O_i \notin C_i$, where $i = 1, 2, \cdots, m$.

Proof. Assume that O_i is an outlier object and $O_i \in C_i$ where C_i is a normal cluster. As per *Definition* 6.5.2, if $O_i \in C_i$, O_i must satisfy the inlier condition w.r.t. τ for class C_i. However, since O_i is an outlier, it contradicts and hence $O_i \notin C_i$. □

The first lemma says that two outlier objects in two distinct neighborhoods will have different ROS' scores with respect to a given cluster The second lemma ensures that our approach is able to unequivocally separate outliers from normal points.

We estimate the outlier score for each candidate data points w r.t the user defined threshold τ for detecting the anomalies or outliers for all cases discussed above. The ranking of the outliers is based on the score; the outlier with the highest score gets the highest rank and vice versa. But in case of compact outlierness, we calculate the average dissimilarity within the cluster as well as compute the total number of points within a cluster. However, the score is calculated by selecting the relevant feature set, i.e., $\{f_1, f_2, \cdots, f_i\}$ to reduce the computational cost. Based on these values, we can detect anomalies or outliers. The proposed technique successfully handles all the defined cases over several real and synthetic datasets. It detects attacks based on the outlier score ROS' from the candidate data objects. The major steps of attack or outlier detection are given in Algorithm 8.

Algorithm 8 : AttackDETECT $(\mathbb{X}_c, \tau)$

Input: $\mathbb{X}_c$ is the candidate objects and τ is the score threshold
Output: O_i as anomalies

1. **for** $i \leftarrow 1$ to $\mathbb{X}_c$ **do** ▷ $c = 1, 2, 3, \cdots n$ are the candidate objects
2. compute $Score[i] \leftarrow ROS'(\mathbb{X}_c.\mu_i, F')$ ▷ score is a single dimensional array to store each score values
3. sort $(Score[i])$ ▷ sort the score values in ascending order
4. **end for**
5. Report the anomalies or outliers, O_i's w r.t. the threshold τ

6.5.5 Empirically Comparing ROS′ and ROS

Using the ROS′ we can estimate the outlier score for each candidate object w.r.t the reference points to identify the nonconforming patterns Reference points are computed based the clusters, $C_1, C_2, C_3 \cdots, C_k$ obtained from TreeCLUS. It generates clusters using subset of relevant features obtained from IGFS algorithm. During score computation, the ROS′ function computes same class occurrences with respect to k' nearest neighbors. In addition, it estimates a summary of distance measures between candidate objects and reference points over a subset of relevant features.

A multiplier factor $\frac{max_{1 \le i \le k'} S_i}{k'}$ is introduced to enhance possible sensitivity in score values of closely spaced objects The ROS and ROS′ score values with spacing between them w.r.t. the sample dataset described in Table 6.2 are illustrated in Table 6.4. As seen in the table, the score values of objects O_3 and O_9 do not vary w.r.t. the change of candidate object in case of ROS. Hence, we claim that our ROS′ improves the sensitivity between any two candidate objects w.r.t. the reference points to identify outliers or anomalies effectively ROS′ has the following major advantages.

- It is sensitive to closely spaced objects in terms of score values.
- It works in subspace and high dimensional large datasets.
- It significantly improves the detection rate especially for Probe, U2R and R2L attacks (see Table 6.18), which are similar to normal traffic.
- Spacing between ROS and the ROS′ score values is high enough to allow one to effectively identify anomalies from network traffic.

Table 6.4: ROS and ROS′ score values with spacing

Object ID	*ROS*	*ROS′*	*Spacing*	*ROS*	*ROS′*	*Spacing*
k'	$k' = 4$			$k' = 5$		
O_1	0 16	7 1105	6 5005	0 6026	5 9056	5 303
O_2	0 6674	0 8456	0 1782	0 7215	0 6822	0 0393
O_3	0 2171	2 4231	2 206	0 2312	2 2196	1 9884
O_4	0 621	3 3158	2 6948	0 6546	2 2234	1 5688
O_5	0 1154	2 8112	2 6958	0 1794	1 8838	1 7044
O_6	0 61	6 4715	5 8615	0 6103	5 5317	4 9214
O_7	0 2465	4 8093	4 5628	0 2601	3 1938	2 9337
O_8	0 6084	3 4627	2 8543	0 638	2 3394	1 7014
O_9	0 2167	2 1123	1 8956	0 2309	2 0299	1 7990
O_{10}	0 0201	1 8459	1 8258	0 0334	0 9801	0 9467
O_{11}	0 1154	2 7054	2 59	0 198	1 7574	1 5594
O_{12}	0 2602	0 4042	0 144	0 2618	0 3144	0 0526
O_{13}	0 1603	5 1807	5 0204	0 1824	4 3574	4 175
O_{14}	0 2541	0 4551	0 201	0 2533	0 3374	0 0841
O_{15}	0 0 0588	2 3396	2 2808	0 0522	1 4851	1 4329
O_{16}	0 6008	2 0588	1 458	0 6611	1 7287	1 0676

The proposed technique can use other proximity measures during cluster formation and score computation A general comparison of Euclidean, Pearson Correlation Coefficient (PCC), Manhattan and cosine distances with arbitrarily selected five objects for score computation with the synthetic zoo, TUIDS KDDcup99, and NSL-KDD datasets is given in Table 6 5 As seen in the table, the score value is not affected very much when the proximity measure is changed Hence, we say that the score computation function is flexible to use any proximity measure

6.5.6 Complexity Analysis

Finding reference points and estimating scores estimation play important roles in the effectiveness and efficiency of our technique The cluster formation algorithm, TreeCLUS generates near balanced tree and takes average $O(nlogk)$ time, where each data object is processed once Here, n is the number of data objects and k is the number of clusters Calculating the reference point and outlier score for each candidate data object takes $O(Nklogk)$ time, where N is the number of candidate data objects, $N > n$, and k is the number of reference points So, the total time complexity of the proposed technique is $O(nlogk) + O(Nklogk)$ A general comparison of the proposed technique with the other similar algorithms is given in Table 6 6

Table 6.5: Comparison of different proximity measures for outlier score computation It is useful for identification of the appropriate proximity measure

Dataset	Scores for five distinct objects			
	Euclidean	*PCC*	*Manhattan*	*Cosine*
Synthetic	0 2361	0 2302	0 2487	0 2210
	4 7192	4 7040	4 7654	4 6825
	0 3281	0 3250	0 3375	0 3163
	3 5676	3 5574	3 6086	3 0755
	4 9650	4 9615	4 9783	4 7501
Zoo	0 0186	0 0179	0 0178	0 0192
	2 1074	2 1069	2 1170	2 1082
	0 2810	0 2906	0 2933	0 2817
	0 3206	0 3169	0 3209	0 3285
	1 1066	1 2048	1 3245	1 2850
TUIDS	0 1134	0 1204	0 1312	0 1322
	0 2122	0 2251	0 2189	0 2137
	2 9231	2 9340	2 9201	2 9103
	2 1503	2 1712	2 1820	2 1289
	3 9120	3 8843	3 8450	3 8930
KDDcup99	0 4520	0 4534	0 4125	0 4510
	2 3511	2 3430	2 3411	2 3440
	0 7719	0 7720	0 7617	0 7660
	0 9452	0 9341	0 9464	0 9518
	3 9005	4 9615	4 9783	4 7501
NSL-KDD	0 1029	0 1234	0 1098	0 1067
	2 5172	2 5140	2 5540	2 4921
	0 6221	0 6289	0 6212	0 6194
	4 0026	4 0112	4 0141	4 0236
	0 3900	0 3890	0 3951	0 3922

Table 6.6: Comparison of the complexity of proposed technique with competitors

Algorithms	*Number of parameters*	*Complexity (approximate)*
LOF [1]	$(k\ MinPts\ M)$	$O(nlogn)$
ORCA [2]	$(k\ n, D)$	$O(n^2)$
ROS [3]	(n, k, R)	$O(Rnlogn)$
Proposed Technique	$(n\ \tau)$	$O(nlogk) + O(Nklogk)$

6.6 Performance Evaluation

The proposed technique was implemented and its performance evaluated using the following environment It was first tested on several real world datasets from the UCI machine learning repository The technique was later tested with both packet and flow level network intrusion datasets generated in our TUIDS testbed [355], discussed in chapter 4 of this thesis Finally, we tested on benchmark network intrusion datasets The proposed technique was implemented on an HP $xw6600$ workstation with Intel Xeon Processor (3 00Ghz) and 4GB RAM *Java*1 6 0 was

used for the implementation in Ubuntu 10 10 (Linux) platform Java was used to facilitate the visualization of detection results easily

6.6.1 Datasets

To evaluate the performance of the proposed technique, we use a synthetic dataset and several real life datasets

Synthetic and UCI ML Repository Datasets

The first dataset we use is a synthetic 2-dimensional dataset that we have generated ourselves It is generated to reflect the several outlier cases illustrated earlier in Figure 6 1 The characteristics of this dataset are given in row 1 of Table 6 7 In this set of experiments we use twenty additional datasets *zoo*, *shuttle*, *breast cancer*, *pima* *vehicle*, *diabetes*, *led7*, *lymphography*, *glass*, *heart*, *hepatitis* *horse*, *ionosphere*, *iris* *sonar*, *waveform* *wine* *lung cancer*, *poker hand* and *abalone* [356] The characteristics of these datasets are also given in Table 6 7

Table 6.7: Characteristics of synthetic and various real life datasets

Serial No	*Dataset*	*Dimensions*	*No of instances*	*No of classes*	*No of Outliers*
1	Synthetic	2	1000	5	40
2	Zoo	18	101	7	17
3	Shuttle	9	14500	3	13
4	Breast cancer	10	699	2	39
5	Pima	8	768	2	15
6	Vehicle	18	846	4	42
7	Diabetes	8	768	2	43
8	Led7	7	3200	9	248
9	Lymphography	18	148	4	6
10	Glass	10	214	6	9
11	Heart	13	270	2	26
12	Hepatitis	20	155	3	32
13	Horse	28	368	2	23
14	Ionosphere	34	351	3	59
15	Iris	4	150	4	27
16	Sonar	60	208	2	90
17	Waveform	21	5000	3	87
18	Wine	13	178	3	45
19	Lung Cancer	57	32	3	9
20	Poker Hand	10	25010	10	16
21	Abalone	8	4177	29	24

Real-life TUIDS Packet and Flow Level Intrusion Datasets

We use real-life packet and flow level feature datasets generated by us using our TUIDS testbed for evaluating the performance of the proposed technique in the

current network scenario. The TUIDS testbed layout, testbed architecture, feature extraction framework, list of extracted features in both packet and flow level have been discussed in Chapter 4. The list of attacks and their characteristics are given in Table 6 8 The characteristics of both packet and flow levels datasets are given in Table 6.9.

Table 6.8: Training and test data attack types with their tools

Serial No	*Attack name*	*Attack generation tool*	*Train dataset*	*Test dataset*
1	Bonk	targa2 c	✓	✓
2	Jolt	targa2 c	✓	✓
3	Land	targa2 c	✓	✓
4	Saihyousen	targa2 c	✓	✓
5	TearDrop	targa2 c	✓	✓
6	Newtear	targa2 c	✓	✓
7	1234	targa2 c	✓	✓
8	Winnuke	targa2 c	✓	✓
9	Oshare	targa2 c	✓	✓
10	Nestea	targa2 c	-	✓
11	SynDrop	targa2 c	✓	✓
12	TcpWindowScan	Nmap	✓	✓
13	SynScan	Nmap	✓	✓
14	XmassTree-Scan	Nmap	-	✓
15	Smurf	smurf4 c	-	✓
16	OpenTear	opentear c	-	✓
17	LinuxICMP	linux-icmp c	-	✓
18	Fraggle	fraggle c	-	✓

Table 6.9: Distribution of normal and attack connection instances in real-life packet and flow level 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	-

Real-life TUIDS Coordinated Scan Datasets

This dataset is built from several scans [5] launched in a coordinated way using the nmap[2] tool on the TUIDS testbed We use the same framework for extracting various features to prepare the coordinated scan datasets at both packet and flow levels. The list of scans and their characteristics at both packet and flow levels are

[2]http //nmap.sourceforge.net/

given in Table 6.10. The distribution of normal and attack data for this dataset at both packet and flow levels is given in Table 6.11.

Table 6.10: Port scan types and firewall level detection possibilities

Port scanning technique	Protocol	TCP flag	Target reply (open port)	Target reply (closed port)	Firewall level detection possibility
TCP *Connect()*	TCP	SYN	ACK	RST	Yes
Reverse Ident	TCP	No	No	No	No
SYN Scan	TCP	SYN	ACK	RST	Yes
IP Header Dump Scan	TCP	No	No	No	No
SYN|ACK Scan	TCP	SYN|ACK	RST	RST	Yes
FIN Scan	TCP	FIN	No	RST	No
ACK Scan	TCP	ACK	No	RST	No
NULL Scan	TCP	No	No	RST	No
XMAS Scan	TCP	All flags	No	RST	No
TCP Fragment	TCP	No	No	No	No
UDP Scan	UDP	No	No	Port Unreachable	No
FTP Bounce Scan	FTP	Arbitrary Flag Set	No	No	No
Ping Scan	ICMP	No	Echo Reply	No	Yes
List Scan	TCP	No	No	No	No
Protocol Scan	IP	No	-	-	No
TCP window scan	TCP	ACK	RST	RST	No

Table 6.11: Distribution of normal and attack connection instances at real-life packet and flow levels for TUIDS coordinated scan datasets

	Dataset type			
Connection type	*Training dataset*		*Testing dataset*	
Packet level				
Normal	65285	90 14%	41095	84 95%
Probe	7140	9 86%	7283	15 05%
Total	72425	-	48378	-
Flow level				
Normal	20180	73 44%	15853	65 52%
Probe	7297	26 56%	8357	34 52%
Total	27477	-	24210	-

KDDcup99 and NSL-KDD Intrusion Datasets

In another set of experiments, we use the well-known KDDcup99 [356] intrusion dataset and an enhanced version of the KDDcup99 dataset known as the NSL-KDD[3] intrusion dataset. The training data contains about five million network connection records. A connection record is a sequence of TCP packets with well defined starting and ending times. Each connection record is unique in the dataset with 41 continuous and nominal features plus one class label. In this work, nominal features such as protocol (e.g., *tcp, udp, icmp*), service type (e.g., *http, ftp, telnet*) and TCP status flags (e.g., *sf, rej*) are converted into numeric features. We convert

[3]http.//www.iscx.ca/NSL-KDD/

categorical attribute values to numeric attribute values by replacing categorical values by numeric values. For example, in the protocol attribute, the value TCP is changed to 1, UDP is changed to 2 and ICMP is changed to 3

Features can be categorized into four types: *basic* features, *content-based* features, *time-based* features and *connection-based* features The categories of the features, labels of the features and their corresponding description are explained in Chapter 4.

The KDDcup99 10% corrected dataset contains 24 attack types categorized into four different groups: DoS (Denial of Service), Probe, R2L (Remote to Local), and U2R (user to Root) attacks The KDDcup99 corrected dataset contains 37 attack types DoS attacks consume computing or memory resources to prevent legitimate behavior of users. Probe is a type of attack where an attacker scans a network to gather information about the target host. In R2L type of attacks, the attacker does not have an account on the victim machine, hence sends a packets to it over a network to illegally gain access as a local user. Finally, in case of U2R attacks, the attacker has local access to the system and is able to exploit vulnerabilities to gain root permissions. The characteristics of KDDcup99 and NSL-KDD intrusion datasets are given in Table 6.12

Table 6.12: Distribution of normal and attack connection instances in both KDDcup99 and NSL-KDD intrusion datasets

Connection type	Dataset type			
	Training dataset		*Testing dataset*	
KDDcup99 dataset				
	10% corrected		Corrected	
Normal	97278	19 69%	60593	19 48%
DoS	391458	79 24%	229853	73 90%
Probe	4107	0 83%	4166	1 34%
R2L	1126	0 22%	16189	5 20%
U2R	52	0 01%	228	0 07%
Total	494021	-	311029	-
NSL-KDD dataset				
Normal	67343	53 46%	9711	43 07%
DoS	45927	36 46%	7460	33 09%
Probe	11656	9 25%	2421	10 74%
R2L	995	0 79%	2753	12 21%
U2R	52	0 04%	199	0 88%
Total	125973	-	22544	-

6.6.2 Experimental Results

The objective of our approach is to accurately classify network traffic data as normal or attack. We measure the accuracy of our approach using precision, recall, F-measure, detection rate and accuracy, described in Chapter 2.

We compare our technique with LOF [1], ORCA [2], ROS [3] and OutRank(b) [357] using benchmark large high dimensional datasets. However, OutRank(b) is tested using synthetic and UCI ML repository datasets only. OutRank(b) is a stochastic graph based outlier detection technique. For OutRank(b), we set the value of the outlierness threshold T to the range 0.68 to 0 89 for the UCI datasets to achieve better results. The rest of the techniques work as follows. LOF is a well known density based outliers detection technique We set $k = 10$ as the distance neighborhood We also set the minimum number of neighboring points $MinPts = 30$ for LOF as suggested in [1] to achieve maximum accuracy. ORCA [2] is a benchmark distance based outlier detection method, which claims to cut down the complexity from $O(n^2)$ to near linear time. The parameters k and N are the number of k nearest neighbors and the number of anomalies needed to report, respectively. We use the reasonable values of $k = 5$ and $N = \frac{n}{8}$, unless otherwise specified although default values are $k = 5$ and $N = 30$. ROS is a reference based outlier detection technique [3] for large datasets. We set the reference based nearest neighbors using $k = 4$ and setting the number of reference points to 18, which is equal to the number of classes in the dataset as recommended in [3] to achieve high accuracy.

Synthetic and UCI ML Repository Datasets

To start, we evaluate the proposed technique using a two dimensional *synthetic* dataset, comprising of 1000 data objects, out of which 4% are outliers. Results of the proposed technique both in terms of detection rate (DR) and false positive rate (FPR) for this dataset are given in the last column of the first row in Table 6.13. Following this, we downloaded executable versions of LOF[4] and ORCA[5] [2]. Results of LOF and ORCA are also given for this dataset in columns 4 and 5, respectively.

[4]http //sites google com/site/rafalba/
[5]http //www.stephenbay.net/orca/

The proposed technique was also evaluated on several other real life benchmark datasets In this work we report our results for *zoo*, *shuttle* *breast cancer*, *pima*, *vehicle*, *diabetes*, *led7* *lymphography*, *glass*, *heart*, *hepatitis*, *horse* *ionosphere*, *iris*, *sonar*, *waveform*, *wine*, *lung cancer* *poker hand* *abalone* datasets and compare them with the three other algorithms The performance of our technique is consistently better than that of the other three algorithms

Table 6.13: Experimental results with Synthetic and UCI ML Repository datasets

Datasets	τ	*Effectiveness*	*LOF* [1]	*ORCA* [2]	*OutRank(b)* [357]	**Proposed Technique**
Synthetic	0 39	DR	0 7500	0 8500	0 8850	1 0000
		FPR	0 0229	0 0166	0 0211	0 0000
Zoo	0 58	DR	0 8235	0 8823	1 0000	0 9411
		FPR	0 1904	0 1309	0 0000	0 0238
Shuttle	0 47	DR	0 8461	0 7692	0 8629	0 9230
		FPR	0 0310	0 0241	0 0208	0 0103
Breast Cancer	0 61	DR	0 8643	0 8109	0 9085	0 9321
		FPR	0 0367	0 0265	0 0183	0 0249
Pima	0 82	DR	0 9333	0 9041	1 0000	1 0000
		FPR	0 0020	0 0211	0 0000	0 0000
Vehicle	0 98	DR	0 3095	0 2919	0 6428	0 7768
		FPR	0 0685	0 0711	0 0354	0 0231
Diabetes	1 9	DR	0 5813	0 5925	0 8139	0 8691
		FPR	0 0385	0 0358	0 0171	0 0198
Led7	0 53	DR	0 2217	0 7310	0 9799	0 9819
		FPR	0 1555	0 0299	0 0040	0 0038
Lymphography	0 44	DR	0 7500	0 7720	1 0000	1 0000
		FPR	0 0074	0 0062	0 0000	0 0000
Glass	0 77	DR	0 8813	0 8488	0 8956	0 9826
		FPR	0 0260	0 0327	0 0227	0 0049
Heart	1 3	DR	0 9108	0 8969	0 8762	0 9928
		FPR	0 0035	0 0107	0 0249	0 0011
Hepatitis	0 77	DR	0 8702	0 8621	0 9183	0 9899
		FPR	0 0247	0 0299	0 0178	0 0010
Horse	0 59	DR	0 9112	0 8822	0 9326	0 9705
		FPR	0 0199	0 0205	0 0118	0 0019
Ionosphere	0 81	DR	0 8108	0 7988	0 8329	0 9523
		FPR	0 0277	0 0312	0 0265	0 0108
Iris	0 43	DR	0 8911	0 8633	0 9013	0 9900
		FPR	0 0211	0 0290	0 0203	0 0001
Sonar	0 66	DR	0 8800	0 8477	0 8551	0 9666
		FPR	0 0201	0 0390	0 0294	0 0110
Waveform	0 79	DR	0 8613	0 8387	0 9112	0 9209
		FPR	0 0260	0 0305	0 184	0 0150
Wine	0 89	DR	0 9233	0 9122	0 9416	1 000
		FPR	0 0166	0 0179	0 0119	0 0000
Lung Cancer	0 38	DR	0 9310	0 8934	0 9717	1 000
		FPR	0 0110	0 0211	0 0108	0 0000
Poker Hand	0 40	DR	0 9620	0 9278	0 9199	0 9911
		FPR	0 0111	0 0190	0 0196	0 0005
Abalone	0 57	DR	0 8902	0 8691	0 8751	0 9890
		FPR	0 0243	0 0380	0 0289	0 0050

Real-life TUIDS Packet and Flow level Intrusion Datasets

We also present results with real-life packet and flow level network intrusion datasets for our technique We convert all categorical attributes into numeric form and then

compute the $log_z(a_{i,j})$ of the larger values for normalization. The value of z depends on the attribute values and $a_{i,j}$ represents the largest attribute values. We use 50% of the dataset for training purpose with normal and DoS attacks, and the remaining 50% of the dataset for testing purpose during performance analysis, as given in Table 6.9. We evaluate in terms of Precision, Recall and F-measure. We provide confusion matrices for LOF [1], ORCA [2], ROS [3] and our technique using both packet and flow levels TUIDS intrusion datasets in Table 6.14. Currently, we analyze only two types of attacks, DoS and probe. The detection rate for both packet and flow level intrusion datasets is better for DoS and probe attack instances than normal instances due to the lack of pure normal instances collected from our testbed. It is still a challenge to obtain pure normal instances from an enterprise networks. It is important to note this because a collection of a large number of pure normal instances is vital in real-life network anomaly detection.

Real-life TUIDS Packet and Flow level Coordinated Scan Datasets

We have generated sixteen types of attacks (see Table 6.10) for coordinated scans. However, in this set of experiments, we consider only four types of scans (i.e., TCP SYN, window, XMAS, and NULL) in the coordinated mode during testing with both packet and flow level datasets. With these datasets, we convert all categorical attributes into numeric form and compute $log_z(a_{i,j})$ to normalize the objects values like before We also use 50% of the dataset for training and rest for testing is this dataset. The confusion matrices of LOF [1], ORCA [2] and ROS [3] with our technique using coordinated scan datasets at both packet and flow levels are given in Table 6.15.

KDDcup99 and NSL-KDD Intrusion Datasets

We discuss experimental results for both KDDcup99 and NSL-KDD intrusion datasets with the proposed technique. With these datasets, we convert all categorical attributes into numeric form and compute $log_z(a_{i,j})$ to normalize the objects values like before. We use the KDDcup99 10% corrected dataset for training and KDDcup99 corrected dataset for testing. For additional experiments, we use the KDDTrain+ dataset for the training and KDDTest+ for the testing, both from

Table 6.14: The Confusion matrix for LOF [1] ORCA [2], ROS [3] and our proposed technique using packet and flow level TUIDS intrusion datasets

	Evaluation measures			Confusion matrix				
Connection type	*Precision*	*Recall*	*F measure*	*Value*	*Normal*	*DoS*	*Probe*	*Total*
			LOF [1]					
Packet level								
Normal	0 8837%	0 9096%	0 8965%	Normal	43569	3527	799	47895
DoS	0 8910%	0 8942%	0 8926%	DoS	3196	27375	42	30613
Probe	0 6419%	0 6523%	0 6471%	Probe	2603	94	5060	7757
Average	0 8055%	0 8187%	0 8104%	Total	49368	30996	5901	86265
Flow level								
Normal	0 9056%	0 9127%	0 9091%	Normal	15307	1378	85	16770
DoS	0 8624%	0 8866%	0 8743%	DoS	1573	12834	68	14475
Probe	0 6612%	0 6749%	0 6679%	Probe	2931	151	6398	9480
Average	0 8097%	0 8247%	0 8171%	Total	19811	14363	6551	40725
			ORCA [2]					
Packet level								
Normal	0 8610%	0 8713%	0 8661%	Normal	41732	5214	949	47895
DoS	0 9007%	0 9128%	0 9067%	DoS	2605	27945	63	30613
Probe	0 8703%	0 8847%	0 8774%	Probe	796	98	6863	7757
Average	0 8773%	0 8896%	0 8834%	Total	45133	33257	7875	86265
Flow level								
Normal	0 8832%	0 8927%	0 8879%	Normal	14971	1674	125	16770
DoS	0 9087%	0 9297%	0 9190%	DoS	978	13458	39	14475
Probe	0 8516%	0 8659%	0 8587%	Probe	1217	54	8209	9480
Average	0 8812%	0 8961%	0 8885%	Total	17166	15186	8373	40725
			ROS [3]					
Packet level								
Normal	0 9218%	0 9453%	0 9334%	Normal	45275	1726	894	47895
DoS	0 9538%	0 9629%	0 9583%	DoS	1131	29479	3	30613
Probe	0 8702%	0 8733%	0 8717%	Probe	961	22	6774	7757
Average	0 9153%	0 9272%	0 9211%	Total	47367	31227	7671	86265
Flow level								
Normal	0 9471%	0 9521%	0 9565%	Normal	15968	785	17	16770
DoS	0 9690%	0 9735%	0 9713%	DoS	382	14066	27	14475
Probe	0 8814%	0 8926%	0 8869%	Probe	994	24	8462	9480
Average	0 9325%	0 9394%	0 9382%	Total	17344	14875	8506	40725
			Proposed Technique					
Packet level								
Normal	0 9607%	0 9854%	0 9728%	Normal	47195	633	67	47895
DoS	0 9977%	0 9964%	0 9971%	DoS	110	30503	0	30613
Probe	0 9627%	0 9796%	0 9711%	Probe	143	15	7599	7757
Average	0 9737%	0 9871%	0 9803%	Total	47448	31151	7666	86265
Flow level								
Normal	0 9745%	0 9868%	0 9806%	Normal	16549	214	7	16770
DoS	0 9844%	0 9969%	0 9906%	DoS	41	14431	3	14475
Probe	0 9790%	0 9806%	0 9798%	Probe	176	8	9296	9480
Average	0 9793%	0 9881%	0 9837%	Total	16766	14653	9306	40725

the NSL-KDD datasets We use the feature selection algorithm to select the best feature subsets for outlier based network anomaly detection The selected feature subsets for the KDDcup99 intrusion datasets are given in Table 6 16 The selected feature subset for the NSL-KDD datasets is also given in Table 6 17 In each table, a row represents a selected subset of features and gives the labels of these important features It is clear that after applying the feature selection algorithm, the size of the feature subset used for each class is greatly reduced Hence, the computation time taken by the proposed technique is substantially less than when full feature

Table 6.15: The Confusion matrix for LOF [1] ORCA [2], ROS [3] and our proposed technique using packet and flow level TUIDS coordinated scan datasets

	Evaluation measures			Confusion matrix			
Connection type	Precision	Recall	F-measure	Value	Normal	Probe	Total
			LOF [1]				
Packet level							
Normal	0.8803%	0.8911%	0.8857%	Normal	36620	4475	41095
Probe	0.8030%	0.8137%	0.8083%	Probe	1357	5926	7283
Average	0.8512%	0.8524%	0.8470%	Total	37977	10401	48378
Flow level							
Normal	0.8773%	0.8829%	0.8801%	Normal	13997	1856	15853
Probe	0.8094%	0.8213%	0.8153%	Probe	1493	6864	8357
Average	0.8434%	0.8521%	0.8477%	Total	15490	8720	24210
			ORCA [2]				
Packet level							
Normal	0.9043%	0.9198%	0.9119%	Normal	37798	3297	41095
Probe	0.8802%	0.8830%	0.8816%	Probe	852	6431	7283
Average	0.8922%	0.9014%	0.8967%	Total	38650	9728	48378
Flow level							
Normal	0.9197%	0.9377%	0.9286%	Normal	14865	988	15853
Probe	0.8801%	0.8939%	0.8869%	Probe	886	7471	8357
Average	0.8999%	0.9158%	0.9078%	Total	15751	8459	24210
			ROS [3]				
Packet level							
Normal	0.9126%	0.9265%	0.9195%	Normal	38078	3017	41095
Probe	0.9013%	0.9161%	0.8096%	Probe	611	6672	7283
Average	0.9069%	0.9213%	0.8645%	Total	38689	9689	48378
Flow level							
Normal	0.9015%	0.9145%	0.9079%	Normal	14498	1355	15853
Probe	0.8671%	0.8877%	0.8773%	Probe	938	7419	8357
Average	0.8843%	0.9011%	0.8926%	Total	15436	8774	24210
			Proposed Technique				
Packet level							
Normal	0.9629%	0.9807%	0.9717%	Normal	40301	794	41095
Probe	0.9674%	0.9781%	0.9727%	Probe	159	7124	7283
Average	0.9652%	0.9794%	0.9722%	Total	40460	7918	48378
Flow level							
Normal	0.9708%	0.9839%	0.9773%	Normal	15598	255	15853
Probe	0.9710%	0.9788%	0.9748%	Probe	177	8180	8357
Average	0.9709%	0.9813%	0.9761%	Total	15775	8435	24210

sets are used We provide confusion matrices for LOF [1], ORCA [2] and ROS [3] and our technique with both the KDDcup99 and the NSL-KDD intrusion datasets in Table 6 18

We also consider all attacks in the case of the KDDcup99 corrected dataset for evaluating the proposed technique It has a total of 37 attack classes and the normal class We give a confusion matrix of the results in Table 6 19 with comparison of results with competing algorithms (i.e , CART [6], CN2 [6] and C4 5 [6])

Parameters Selection

The determination of suitable values for parameters plays an important role in any outlier or network anomaly detection method In our approach, ξ is used as a threshold for finding an optimal subset of relevant features for a particular

Table 6.16: Selected relevant features for all classes in the KDDcup99 intrusion dataset

Method	#Features	Selected features
Normal class		
IGFS	10	Src-bytes Service Count Dst-bytes, Dst-host-same-src-port-rate, Srv-count Logged-in, Protocol-type Dst-host-diff-srv-rate, Dst-host-same-srv-rate
FFSA [81]	6	Src-bytes Service, Duration Flag, Dst-host-same-srv-rate, Dst-bytes
MMIFS [81]	6	Src-bytes Count, Service, Dst-bytes, Dst-host-diff-srv-rate Duration
LCFS [81]	15	Logged-in, Dst-host-same-srv-rate, Dst-host-srv-count, Service, Count, Rerror-rate Same-srv-rate, Dst-host-rerror-rate, Dst-host-srv-serror-rate, Srv-rerror-rate Protocol-type, Dst-host-srv-rerror-rate, Srv-serror-rate, Dst-host-diff-srv-rate, Hot
DoS class		
IGFS	12	Src-bytes Count, Service, Dst-host-same-src-port-rate, Dst-host-diff-srv-rate, Srv-count, Dst-host-srv-count, Dst-host-same-srv-rate, Dst-host-serror-rate, Protocol-type, Dst-host-srv-serror-rate, Serror-rate
FFSA [81]	8	Src-bytes Dst-host-serror-rate, Service
MMIFS [81]	8	Src-bytes Count, Dst-bytes, Protocol-type, Srv-count, Dst-host-srv-rerror-rate, Dst-host-same-src-port-rate, Service
LCFS [81]	36	Dst-host-count, Rerror-rate, Count Dst-host-serror-rate, Dst-host-rerror-rate Srv-count Num-compromised, Protocol-type, Dst-host-rerror-rate, Is-guest-login, Diff-srv-rate Serror-rate, Srv-rerror-rate, Dst-host-diff-srv-rate, Srv-serror-rate, Dst-host-srv-diff-host-rate Logged-in, Dst-host-same-src-port-rate, Dst-host-srv-serror-rate Duration, Hot Root-shell, Num-failed-logins, Num-file-creations, Dst-host-srv-count, Num-root, Num-access-files, Num-shells, Urgen, Src-bytes, Dst-host-same-srv-rate Srv-diff-host-rate, Dst-bytes, Service, Same-srv-rate
Probe class		
IGFS	14	Service, Src-bytes, Rerror-rate, Count, Dst-host-srv-diff-host-rate, Flag, Dst-host-rerror-rate, Dst-host-same-src-port-rate, Dst-host-count, Dst-host-same-srv-rate Dst-host-diff-srv-rate, Dst-host-srv-count, Same-srv-rate, Dst-bytes
FFSA [81]	24	Dst-host-rerror-rate, Src-bytes, Dst-host-srv-rerror-rate, Num-failed-logins Protocol-type, Is-guest-login, Urgen, Rerror-rate, Dst-host-srv-diff-host-rate, Srv-rerror-rate, Root-shell, Num-access-files, Srv-diff-host-rate, Num-shells Duration, Num-file-creations, Num-root, Num-compromised, Serror-rate, Dst-host-srv-serror-rate Srv-serror-rate, Dst-bytes, Diff-srv-rate, Dst-host-count
MMIFS [81]	13	Dst-host-rerror-rate, Src-bytes, Dst-host-srv-count, Count, Srv-rerror-rate, Service, Dst-host-srv-rerror-rate, Num-compromised, Rerror-rate, Dst-host-count, Logged-in, Srv-count Srv-rerror-rate
LCFS [81]	7	Rerror-rate, Logged-in, Dst-host-rerror-rate Dst-host-srv-rerror-rate, Dst-host-same-srv-rate, Srv-rerror-rate Dst-host-diff-srv-rate
R2L class		
IGFS	13	Service, Src bytes, Dst-bytes, Dst-host-srv-count Count, Dst-host-same-src-port-rate Dst-host-srv-diff-host-rate Srv-count, Dst-host-count Flag, Dst-host-srv-serror-rate Dst-host-diff-srv-rate, Dst-host-serror-rate
FFSA [81]	10	Service, Dst-bytes Flag Num-failed-logins, Urgen, Dst-host-srv-count, Dst-host-srv-diff-host-rate, Dst-host-serror-rate Is-guest-login, Serror-rate
MMIFS [81]	15	Service, Num-compromised Is-guest-login Count, Hot, Src-bytes, Dst-host-diff-srv-rate, Srv-count, Dst-bytes Dst-host-srv-count, Dst-host-srv-diff-host-rate, Dst-host-count, Duration, Dst-host-srv-diff-host-rate, Dst-host-srv-serror-rate Is-guest-login, Dst-host-serror-rate, Hot Service
LCFS [81]	4	Is-guest-login, Dst-host-serror-rate, Hot, Service
U2R class		
IGFS	15	Service, Dst-host-srv-count Duration, Src-bytes, Num-file-creations, Root-shell, Hot, Dst-host-count, Num-compromised Dst-host-same-src-port-rate, Srv-count, Dst-host-diff-srv-rate, Dst-host-same-srv-rate, Dst-host-srv-diff-host-rate, Count
FFSA [81]	27	Src-bytes Duration, Num-access-files, Num-shells, Dst-host-srv-serror-rate Protocol-type, Is-guest-login, Urgen, Same-srv-rate Land Wrong-fragment, Su-attempted, Diff-srv-rate, Num-root, Num-outbound-cmds, Is-host-login, Dst-bytes, Service, Srv-serror-rate Srv-diff-host-rate, Dst-host-srv-count, Root-shell, Flag, Num-file-creations, Dst-host-count Logged-in, Serror-rate
MMIFS [81]	10	Src-bytes Duration, Service Srv-count, Count, Protocol-type Dst-host-srv-count, Dst-bytes Dst-host-count, Flag, Root-shell Is-host-login
LCFS [81]	3	Root-shell, Num-file-creations, Num-compromised

class in a dataset. The optimality of a feature subset is decided based on the classification accuracy obtained using a dataset For our sample dataset, we observe experimentally that for $\xi \geq 0.917$ the best possible accuracy is obtained However,

Table 6.17: Selected relevant features for all classes in the NSL-KDD intrusion dataset

Method	#Features	Selected features
Normal	11	Src-bytes Service, Dst-bytes Flag Diff-srv-rate, Same-srv-rate, Dst-host-srv-count, Dst-host-same-srv-rate, Dst-host-diff-srv-rate, Dst-host-serror-rate, Logged-in
DoS	9	Src-bytes Count Dst-bytes, Srv-count Dst-host-same-src-port-rate, Dst-host-same-srv-rate, Dst-host-srv-serror-rate, Protocol-type, Serror-rate
Probe	13	Src-bytes Count, Dst-host-rerror-rate, Flag Num-failed-logins, Is-guest-login, Dst-host-same-src-port-rate, Dst-host-diff-srv-rate Dst-host-srv-count, Same-srv-rate. Dst-bytes Diff-srv-rate, Dst-host-serror-rate
R2L	10	Service, Dst-bytes Src-bytes, Dst-host-srv-count, Dst-host-srv-diff-host-rate Srv-count, Flag, Is-guest-login, Dst-host-srv-serror-rate, Dst-host-diff-srv-rate
U2R	16	Src-bytes Duration, Service, Dst-host-srv-count, Root-shell, Urgen, Same-srv-rate, Land Wrong-fragment, Dst-host-same-src-port-rate, Srv-count, Num-root, Num-outbound-cmds, Is-host-login, Dst-host-srv-diff-host-rate Count

for other datasets it differs. In TreeCLUS, the value of the parameter α' is selected using a heuristic approach. $\alpha' \leq 10.5$ initially for the sample dataset, but it is different for other datasets β' is the number of data objects needed to satisfy the neighborhood condition over a subset of features to form a node (i.e., $\beta' = 2$ in case of our sample dataset). Its value is the same for all datasets. Finally, in the outlier detection algorithm, k' and τ are two important parameters. If k' and τ are not properly selected, it may affect the accuracy of the detection method. We select both these parameter values using a heuristic approach. We find k' values for different datasets heuristically as shown in Figure 6.6. We find the best possible solution for k' values ranging from 22 to 47.

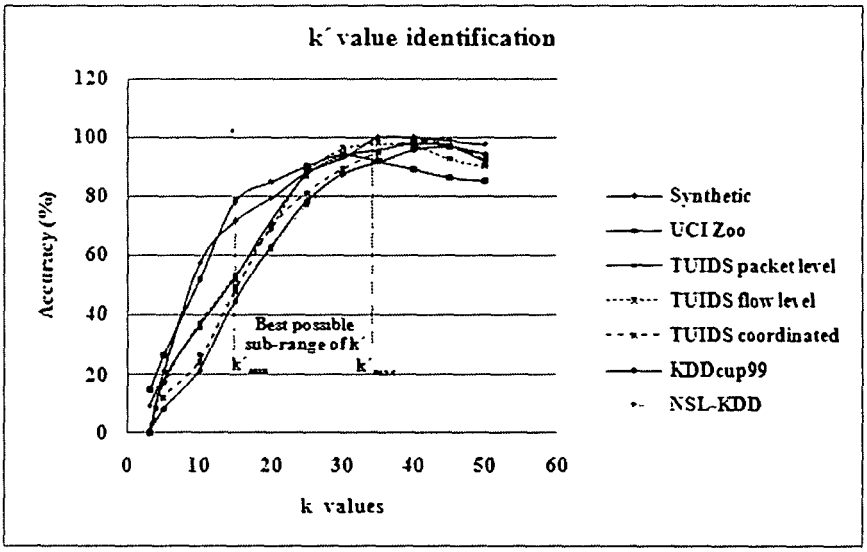


Figure 6.6: k' values vs. accuracy for the identification of k' values. k'_{min} is the minimum range of k' values and k'_{max} is the maximum range of k' values. It is useful for selecting k' values during score computation.

Table 6.18: The Confusion matrices for LOF [1], ORCA [2], ROS [3] and our proposed technique using KDDcup99 and NSL-KDD intrusion datasets

Evaluation measures				Confusion matrix						
Connection type	*Precision*	*Recall*	*F measure*	*Value*	*Normal*	*R2L*	*DoS*	*Probe*	*U2R*	*Total*
LOF [1]										
KDDcup99 dataset										
Normal	0.8902%	0.9096%	0.8998%	Normal	55119	5294	163	15	2	60593
R2L	0.7164%	0.7283%	0.7223%	R2L	4165	11791	14	216	3	16189
DoS	0.8733%	0.8929%	0.8829%	DoS	22412	17	205258	2164	2	229853
Probe	0.8399%	0.8550%	0.8474%	Probe	493	20	91	3562	0	4166
U2R	0.6092%	0.6140%	0.6115%	U2R	75	10	2	1	140	228
Average	0.7858%	0.7999%	0.7928%	Total	82264	17132	205528	5958	147	311029
NSL-KDD dataset										
Normal	0.9186%	0.9314%	0.9249%	Normal	9045	480	124	56	6	9711
R2L	0.6897%	0.7043%	0.6969%	R2L	770	1939	0	43	1	2753
DoS	0.8611%	0.8752%	0.8681%	DoS	792	37	6529	94	8	7460
Probe	0.8549%	0.8612%	0.8580%	Probe	39	7	287	2085	3	2421
U2R	0.6107%	0.6231%	0.6168%	U2R	64	8	3	0	124	199
Average	0.7870%	0.7990%	0.7929%	Total	10710	2471	6943	2278	142	22544
ORCA [2]										
KDDcup99 dataset										
Normal	0.9272%	0.9389%	0.9330%	Normal	56896	3125	486	81	5	60593
R2L	0.7219%	0.7406%	0.7311%	R2L	4105	11991	13	79	1	16189
DoS	0.9106%	0.9198%	0.9152%	DoS	18177	0	209799	1877	0	229853
Probe	0.8530%	0.8826%	0.8675%	Probe	341	19	129	3677	0	4166
U2R	0.6197%	0.6315%	0.6255%	U2R	73	7	4	0	144	228
Average	0.8017%	0.8162%	0.8225%	Total	79592	15142	210431	5714	150	311029
NSL-KDD dataset										
Normal	0.9187%	0.9293%	0.9239%	Normal	9024	507	117	58	5	9711
R2L	0.7412%	0.7526%	0.8153%	R2L	617	2072	9	55	0	2753
DoS	0.8951%	0.9089%	0.9019%	DoS	476	5	6781	198	0	7460
Probe	0.8603%	0.8823%	0.8712%	Probe	79	7	198	2136	1	2421
U2R	0.5930%	0.6080%	0.6004%	U2R	69	8	1	0	121	199
Average	0.8189%	0.8328%	0.8413%	Total	10692	2423	7038	2250	141	22544
ROS [3]										
KDDcup99 dataset										
Normal	0.9233%	0.9416%	0.9165%	Normal	57059	3110	366	53	5	60593
R2L	0.6876%	0.6992%	0.6934%	R2L	4813	11320	9	47	0	16189
DoS	0.9004%	0.9011%	0.9007%	DoS	22722	5	207123	3	0	229853
Probe	0.8824%	0.8951%	0.8887%	Probe	339	13	83	3729	2	4166
U2R	0.5928%	0.6008%	0.5967%	U2R	78	11	2	0	137	228
Average	0.7973%	0.8076%	0.7992%	Total	85011	14459	207583	3832	144	311029
NSL-KDD dataset										
Normal	0.9406%	0.9562%	0.9483%	Normal	9286	380	41	2	2	9711
R2L	0.7236%	0.7304%	0.7269%	R2L	686	2011	6	49	1	2753
DoS	0.9128%	0.9214%	0.9170%	DoS	567	17	6874	0	2	7460
Probe	0.8872%	0.9083%	0.8938%	Probe	97	7	113	2199	5	2421
U2R	0.6471%	0.6583%	0.6527%	U2R	56	8	4	0	131	199
Average	0.8222%	0.8349%	0.8277%	Total	10065	2752	7360	2226	141	22544
Proposed Technique										
KDDcup99 dataset										
Normal	0.9863%	0.9983%	0.9769%	Normal	60489	86	13	4	1	60593
R2L	0.8776%	0.8996%	0.8885%	R2L	1596	14563	9	21	0	16189
DoS	0.9988%	0.9999%	0.9994%	DoS	17	0	229833	3	0	229853
Probe	0.9670%	0.9807%	0.9687%	Probe	59	4	17	4086	0	4166
U2R	0.7342%	0.7632%	0.7215%	U2R	41	7	5	1	174	228
Average	0.9128%	0.9283%	0.9110%	Total	62202	14660	229877	4115	175	311029
NSL-KDD dataset										
Normal	0.9801%	0.9902%	0.9851%	Normal	9616	78	13	3	1	9711
R2L	0.8790%	0.8892%	0.8841%	R2L	292	2448	2	11	0	2753
DoS	0.9896%	0.9988%	0.9942%	DoS	9	0	7451	0	0	7460
Probe	0.9690%	0.9798%	0.9744%	Probe	22	1	26	2372	0	2421
U2R	0.7254%	0.7788%	0.7512%	U2R	33	8	3	0	155	199
Average	0.9086%	0.9274%	0.9178%	Total	9972	2535	7495	2386	156	22544

We have analyzed the effect of the threshold τ using synthetic as well as real

Table 6.19: Comparison between CART CN2 C4 5 and Proposed Technique while considering all attacks over KDDcup99 intrusion dataset

Algorithm	CART [6]		CN2 [6]		C4 5 [6]		Proposed Technique	
Connection type	1 Precision	Recall	1 Precision	Recall	1- Precision	Recall	Precision	Recall
normal	0 0540%	0 9430%	0 1442%	0 9822%	0 0507%	0 9436%	0 9563%	0 9938%
snmpgetattack	0 3862%	0 6360%	0 0000%	0 0026%	0 3828%	0 0026%	0 0186%	0 4235%
named	1 0000%	0 0000%	1 0000%	0 0000%	0 5000%	0 2353%	0 6149%	0 7259%
xlock	1 0000%	0 0000%	1 0000%	0 0000%	1 0000%	0 0000%	1 0000%	0 7102%
smurf	0 0000%	1 0000%	0 0011%	1 0000%	0 0000%	1 0000%	1 0000%	1 0000%
ipsweep	0 0199%	0 9641%	0 0752%	0 9248%	0 0000%	0 9902%	0 9902%	0 9967%
multihop	1 0000%	0 0000%	1 0000%	0 0000%	0 0000%	0 0556%	0 8392%	0 6209%
xsnoop	1 0000%	0 0000%	1 0000%	0 0000%	1 0000%	0 0000%	0 7610%	0 7619%
sendmail	1 0000%	0 0000%	1 0000%	0 0000%	1 0000%	0 0000%	1 0000%	0 6622%
guess_passwd	0 0566%	0 9968%	0 0270%	0 9725%	0 0242%	0 9863%	0 9981%	0 9996%
saint	0 0000%	0 1236%	0 2209%	0 8003%	0 2382%	0 8302%	0 3899%	0 9415%
buffer_overflow	1 0000%	0 0000%	1 0000%	0 0000%	0 4118%	0 4545%	1 0000%	0 9605%
portsweep	0 1111%	0 8362%	0 1376%	0 7260%	0 1604%	0 9463%	0 8835%	0 9819%
pod	0 0000%	0 8391%	0 0000%	0 8391%	0 4082%	1 0000%	1 0000%	0 9425%
apache2	1 0000%	0 0000%	0 0399%	0 8476%	0 1841%	0 9937%	1 0000%	0 9962%
phf	1 0000%	0 0000%	1 0000%	0 0000%	1 0000%	0 0000%	1 0000%	1 0000%
udpstorm	1 0000%	0 0000%	1 0000%	0 0000%	1 0000%	0 0000%	1 0000%	1 0000%
warezmaster	0 0137%	0 9863%	0 1428%	0 8546%	0 0293%	0 9944%	0 9972%	0 9714%
perl	1 0000%	0 0000%	1 0000%	0 0000%	1 0000%	0 0000%	1 0000%	1 0000%
satan	0 2917%	0 9724%	0 1113%	0 8898%	0 0628%	0 8598%	0 9177%	0 6328%
xterm	1 0000%	0 0000%	1 0000%	0 0000%	0 0000%	0 2308%	1 0000%	0 7812%
mscan	0 0404%	0 9706%	0 0268%	0 8965%	0 0389%	0 8927%	1 0000%	0 9983%
processtable	0 0250%	0 9750%	0 1086%	0 8762%	0 0000%	0 9789%	1 0000%	1 0000%
ps	1 0000%	0 0000%	1 0000%	0 0000%	1 0000%	0 0000%	1 0000%	0 8290%
nmap	1 0000%	0 0000%	0 0000%	1 0000%	0 3197%	0 9881%	1 0000%	1 0000%
rootkit	1 0000%	0 0000%	1 0000%	0 0000%	0 0000%	0 0769%	0 8241%	0 3940%
neptune	0 0016%	0 9990%	0 0011%	0 9994%	0 0011%	0 9978%	1 0000%	1 0000%
loadmodule	1 0000%	0 0000%	1 0000%	0 0000%	0 0000%	0 5000%	1 0000%	1 0000%
imap	1 0000%	0 0000%	1 0000%	0 0000%	1 0000%	0 0000%	1 0000%	1 0000%
back	0 4583%	1 0000%	0 1164%	0 7951%	0 0132%	0 9545%	1 0000%	1 0000%
httptunnel	0 5088%	0 7025%	0 1744%	0 8987%	0 3553%	0 8038%	1 0000%	0 9701%
worm	1 0000%	0 0000%	1 0000%	0 0000%	1 0000%	0 0000%	0 0000%	0 0000%
mailbomb	0 0000%	0 9516%	0 0161%	0 9998%	0 0062%	0 9982%	0 9996%	0 9989%
ftp_write	1 0000%	0 0000%	1 0000%	0 0000%	1 0000%	0 0000%	1 0000%	1 0000%
teardrop	1 0000%	0 0000%	1 0000%	0 0000%	1 0000%	0 0000%	1 0000%	0 4167%
land	1 0000%	0 0000%	1 0000%	0 0000%	1 0000%	0 0000%	1 0000%	1 0000%
sqlattack	1 0000%	0 0000%	1 0000%	0 0000%	1 0000%	0 0000%	1 0000%	0 5000%
snmpguess	0 0004%	0 9909%	0 3812%	0 3603%	0 0144%	0 9983%	1 0000%	0 9599%
Average	**0 6044%**	**0 3918%**	**0 5518%**	**0 4123%**	**0 4264%**	**0 4924%**	**0 8997%**	**0 8466%**

life datasets (i e *zoo*, *shuttle*, and *breast cancer*) The performance of the proposed technique in terms of detection rate largely depends on the selection of the value of τ, as seen in Figure 6 7 The value of τ is dependent on the dataset used for evaluation This is because each dataset is different from others in terms of attribute values and dimensions So, the threshold differs from dataset to dataset for the best results However, a most probable range of τ values for these datasets is shown with vertically drawn dashed lines in the Figure 6 7 In our experiments, better results are obtained with τ values in the range of (0 30 - 0 54) for the *synthetic* dataset (0 26 - 0 69) for the *zoo* dataset, (0 38 - 0 57) for the *shuttle* dataset and (0 29 - 0 68) for the *breast cancer* dataset This estimation is helpful in choosing the threshold

value τ for experiments

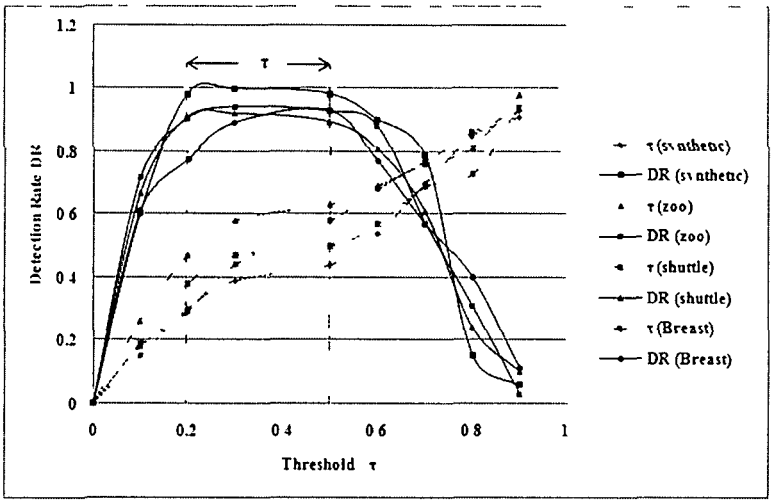


Figure 6.7: Detection rate for different threshold values It is useful for the identification of threshold τ range values, where it performs well

The performance of the proposed technique for an intrusion dataset in terms of precision again largely depends on the selection of τ value as seen in Figure 6 8 The probable range of τ values for each class of attack as well as normal data objects for good results are shown with vertical dashed lines in Figure 6 8 In our experiment we found that good results are obtained for τ values in the range of (0 9 - 2 3) for normal records and (0 4 - 1 15) for attack records

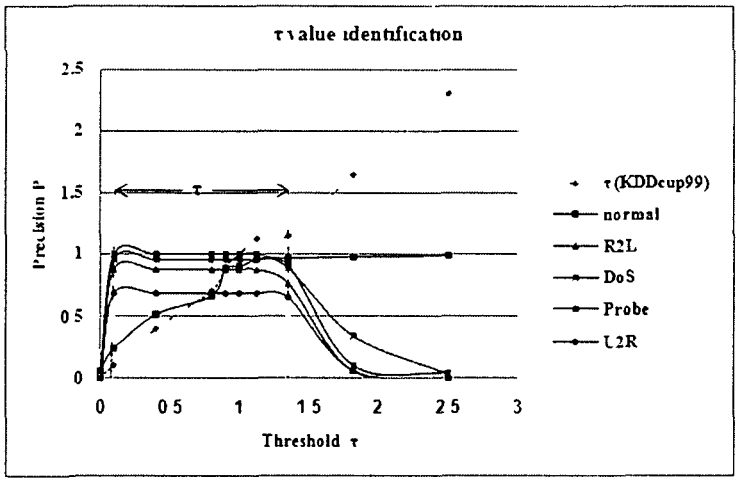


Figure 6.8: Precision for different threshold values It is helpful to identify the threshold τ range values, where it performs best

Discussion

We provide the confusion matrices for LOF, ORCA. ROS, CART, CN2 and C4.5 using several real life network intrusion datasets A performance comparison of the proposed technique with LOF, ORCA and ROS using (a) TUIDS packet level, (b) TUIDS flow level, (c) TUIDS coordinated scan packet and flow level, (d) KDDcup99 and (e) NSL-KDD intrusion datasets is given in Figure 6 10. As seen from the figure, the performance significantly increases for mostly all datasets. We also provide a comparison of the proposed technique using the KDDcup99 intrusion dataset with C4.5, ID3, CN2, CBUID, TANN and HC-SVM in Table 6.20 and Figure 6.9. As seen in the table, the detection rate of normal and U2R instances using our approach is significantly higher those obtained with competing algorithms. DoS and probe attack detection rates are not significantly higher but are better. For R2L attacks, an average detection rate is obtained and it is still better than those obtained by the competing algorithms. Normal, DoS and R2L attack instances are identified with higher detection rate when it was analyzed as individual attack instances.

Table 6.20: Comparison of the proposed technique with other techniques over KDD-cup99 intrusion dataset

Connection type	C4 5 [6]	ID3 [6]	CN2 [6]	CBUID [7]	TANN [8]	HC-SVM [4]	**Proposed Technique**
Normal	94 42%	87 48%	87 08%	-	97 01%	99 29%	99 83%
R2L	81 53%	96 23%	84 51%	8 67%	80 53%	28 81%	89 96%
DoS	99 97%	99 86%	99 93%	99 15%	90 94%	99 53%	99 99%
Probe	94 82%	95 54%	95 85%	80 27%	94 89%	97 55%	98 07%
U2R	67 11%	54 82%	67 54%	60 78%	60 00%	19 73%	76 32%
Average	87 57%	86 78%	86 98%	-	84 67%	68 92%	92 83%

Finally, we present a comparison of the execution time of the proposed technique with the time required by LOF, ORCA and ROS using the KDDcup99 intrusion dataset, in Figure 6.11 The preprocessing time has been excluded in all methods. As seen from the figure, time required by LOF and ORCA increases as the dataset size increases. But ROS and our proposed technique take almost the same time after excluding training time. The proposed technique takes less time than the LOF and the ORCA

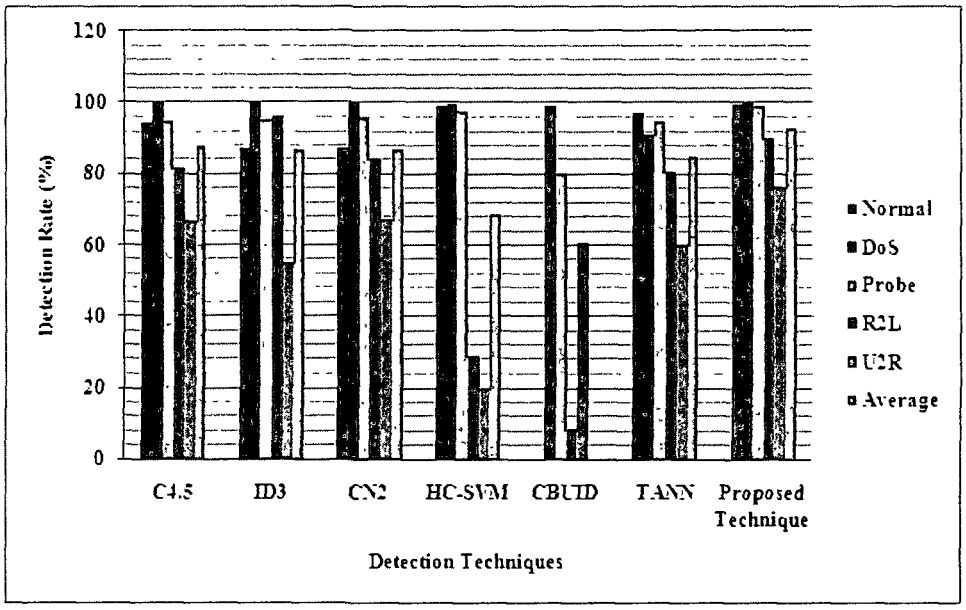


Figure 6.9: A comparison of the proposed technique with C4.5 [6], ID3 [6], CN2 [6], CBUID [7], TANN [8] and HC-SVM [4] using KDDcup99 intrusion dataset

6.7 Summary

In this chapter, an efficient outlier detection technique based on [3] with an application to network anomaly detection is presented. We also present a tree-based subspace clustering algorithm for high dimensional datasets. The clustering algorithm generates the tree in a depth first manner before applying our network anomaly detection algorithm. The main attraction of our technique is its ability to successfully detect all outlier cases. It can also use any proximity measure for score computation. It is important to choose the threshold correctly during network anomaly identification. A heuristic technique is presented for the identification of the threshold. The proposed technique was evaluated with various datasets, viz., (a) synthetic, (b) UCI ML repository datasets, (c) real-life TUIDS intrusion datasets (packet and flow levels), (d) real-life TUIDS coordinated scan datasets (packet and flow levels), and (e) KDDcup99 and NSL-KDD datasets. We compare the performance of our proposed technique with that of other well known outlier detection methods, viz., LOF, ORCA, ROS and also compare it with C4.5, ID3, CN2, CBUID, TANN and HC-SVM, and achieve better performance in almost all the datasets in identifying network anomalies. Hence, we claim that the proposed technique is better than competing algorithms for the intended purpose of network

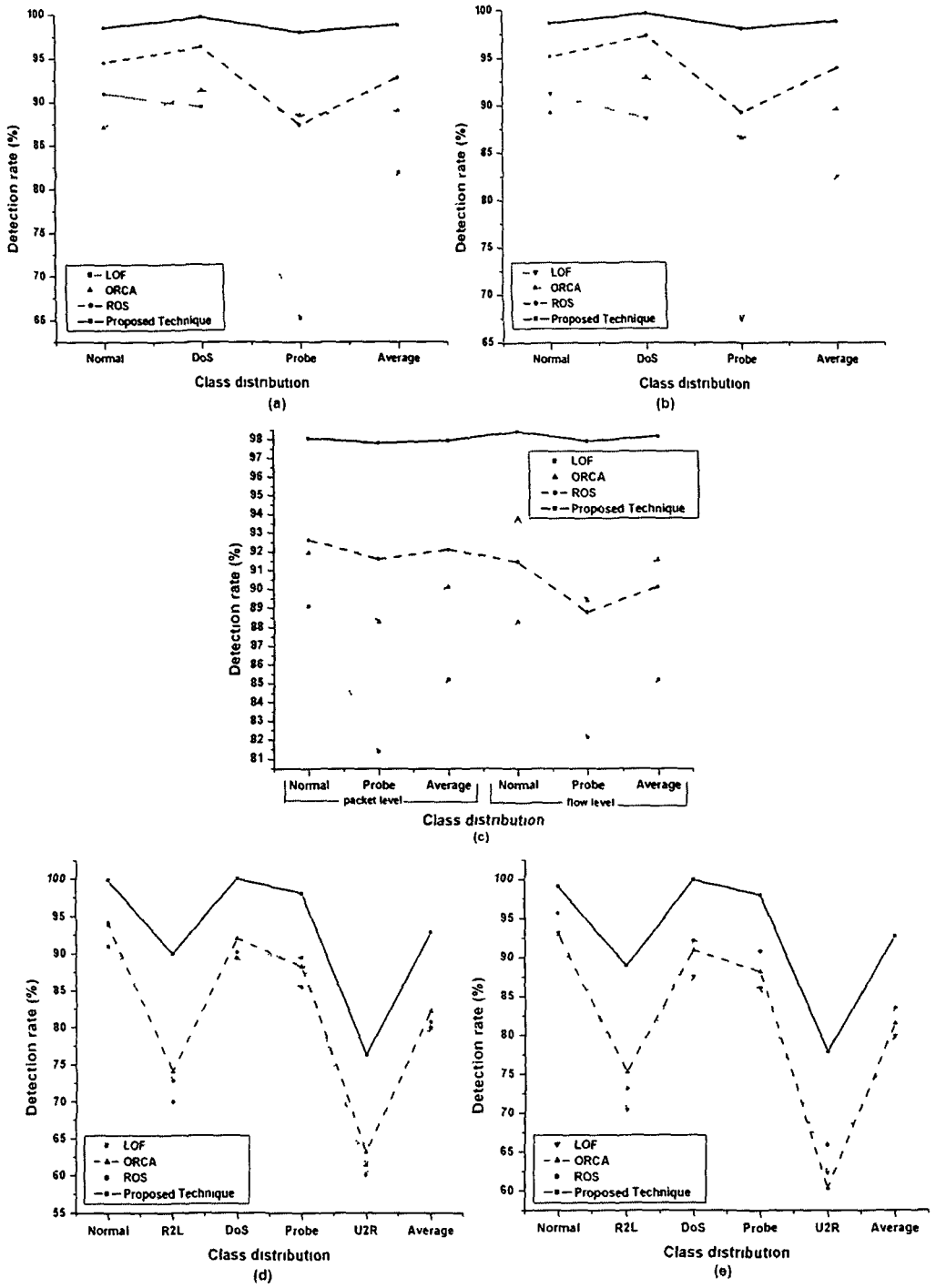


Figure 6.10: A performance comparison of the proposed technique with other techniques using (a) TUIDS packet level (b) TUIDS flow level, (c) TUIDS coordinated scan packet and flow level (d) KDDcup99 and (e) NSL-KDD intrusion datasets

anomaly detection The proposed technique is able to identify anomalies within a 5 second time window There are some published methods which can perform such

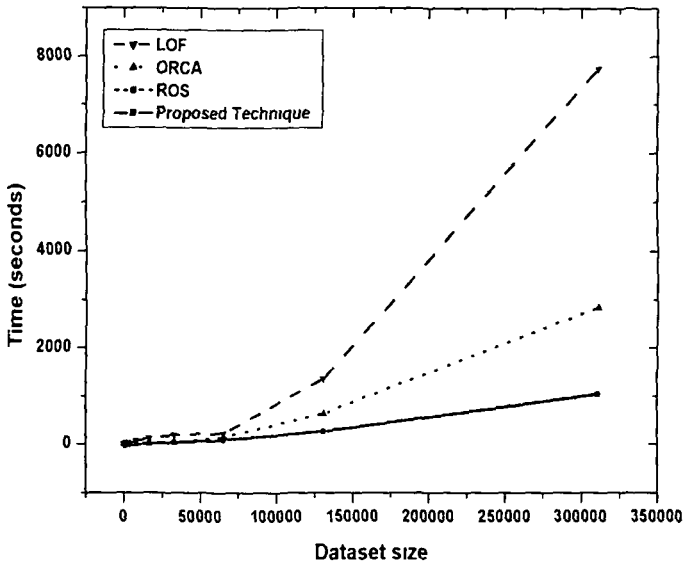


Figure 6.11: An execution time comparison of the proposed technique with LOF [1], ORCA [2] and ROS [3] algorithms based on randomly selected network intrusion dataset size

identification in 2 second time windows, but their detection rate is worse than ours. However, the proposed technique has two limitations. (a) It is dependent on proper tuning of τ w r.t a dataset; we have presented a heuristic method for the selection of τ value. (b) It does not work directly on categorical and mixed types data

Chapter 7

Unsupervised Approach for Network Anomaly Detection

This chapter presents an unsupervised tree-based subspace clustering technique (TreeCLUSS) for finding clusters in network intrusion data and for detecting known as well as unknown attacks without using any labelled traffic or signatures or training. To establish its effectiveness in finding the appropriate number of clusters, we perform a cluster stability analysis. We also introduce an effective cluster labelling technique (CLUSSLab) to label each cluster based on the stable cluster set obtained from TreeCLUSS. CLUSSLab is a multi-objective technique that employs an ensemble approach for labelling each stable cluster generated by TreeCLUSS to achieve a high detection rate. We also introduce an effective unsupervised feature clustering technique to identify a dominating feature subset from each cluster. We evaluate the performance of both TreeCLUSS and CLUSSLab using several real world intrusion datasets to identify known as well as unknown attacks and found that results are excellent.

7.1 Introduction

Advances in networking technology have enabled us to connect distant corners of the globe through the Internet for sharing vast amounts of information. However, along with this advancement, the threat from spammers, attackers and criminal enterprises is also growing at multiple speed [11]. As a result, security experts

use intrusion detection technology to keep secure large enterprise infrastructures Intrusion detection systems (IDSs) are divided into two broad categories· misuse detection [21] and anomaly detection [22] systems Misuse detection can detect only known attacks based on available signatures Thus, dynamic signature updation is important and therefore, new attack definitions are frequently released by IDS vendors However, misuse based systems cannot incorporate or even most all of the rapidly growing number of vulnerabilities and exploits On the other hand, anomaly based detection systems are designed to capture any deviation from profiles of normal behavior. They are more suitable than misuse detection systems for detecting unknown or novel attacks without any prior knowledge However, they normally generate a large number of false alarms.

There are three commonly used approaches for detecting intrusions [12,358] (a) supervised (i.e., both normal and attack instances are used for training), (b) semi-supervised (i.e., only normal instances are used for training) and (c) unsupervised (i.e., without using any prior knowledge). The first two cases require training on the instances for finding anomalies But getting a large amount of labelled normal and attack training instances may not be feasible for a particular scenario. In addition, generating a set of true normal instances with all the variations is an extremely difficult task Hence, unsupervised network anomaly detection, which does not require any prior knowledge of network traffic instances, is more suitable in this situation.

7.1.1 Motivation and Contributions

To overcome obstacles faced by supervised and semi-supervised network anomaly detection methods, unsupervised network anomaly detection methods aim to detect known as well as unknown intrusions without using any prior knowledge of existing network traffic instances. Clustering is an established unsupervised network anomaly detection technique that can be used to identify unknown attacks. However, a common limitation of some clustering approaches is that they require the number of clusters a priori, which often can be difficult to provide. In such cases, stability analysis of the cluster results can be of great help. Validity of the cluster results in terms of real life and benchmark datasets is important to establish the

effectiveness of the results. In high dimensional data, many features are irrelevant to form a specific set of clusters when a full space clustering technique is applied. These are the reasons why we develop an unsupervised method for identification of known and unknown attacks with minimum false alarms.

We aim to provide an unsupervised solution for identifying network attacks with high detection rate. The main contributions of this chapter are stated below.

- We introduce a tree based clustering technique (TreeCLUSS) to identify network anomalies in high dimensional datasets The following are some of the advantages of the proposed TreeCLUSS algorithm.
  - The number of clusters is not required as input parameters
  - It is free from the use of a specific proximity measure.
  - It requires a minimum number of input parameters and the results are not heavily dependent on them
  - It is able to identify both known as well as unknown attacks
- We present a cluster stability analysis to obtain a stable set of results generated by TreeCLUSS. It uses majority voting based decision for cluster stability to get a stable set of clusters.
- We introduce a cluster labelling technique (CLUSSLab) for labelling the clusters generated by TreeCLUSS as normal or attack. It uses a majority voting based decision fusion technique of the results of various cluster indices, cluster sizes and dominating features sets.
- Finally, we develop an effective unsupervised feature clustering technique to identify a dominating feature subset for each stable cluster that is used for cluster labelling. It is important to identify a relevant feature set for a particular set of clusters to match with a previously identified feature set during cluster labelling.

7.2 Prior Research

The problem of unsupervised detection of network attacks and intrusions has been studied for many years with the goal of identifying unknown attacks in high speed network traffic data. Most network based intrusion detection systems (NIDSs) are misuse or signature based. For example, SNORT [359] and BRO [360] are two well-known open source misuse based NIDS. To overcome the inability of such systems to detect unknown attacks, novel anomaly based NIDSs have been introduced in the past decade. A detailed study can be found in [119, 123]. Here, we briefly discuss some recent unsupervised network anomaly detection methods.

7.2.1 Clustering-based Network Anomaly Detection

Clustering is an important technique used in unsupervised network intrusion detection. A majority of unsupervised network anomaly detection techniques are based on clustering and outlier detection [5, 64, 164]. Leung and Leckie report a grid based clustering algorithm to achieve reduced computational complexity [164]. An unsupervised intrusion detection method by computing cluster radius threshold (CBUID) is proposed by [7]. The authors claim that CBUID works in linear time with respect to the size of datasets and the number of features. Song et al. report an unsupervised auto-tuned clustering approach that optimizes parameters and detects changes based on unsupervised anomaly detection for identifying unknown attacks [29]. Noto et al. present a new semi-supervised anomaly detection method (FRaC) [30] that builds an ensemble of feature models based on normal instances, and then identifies instances that disagree with these models as anomalous. Casas et al. present a novel unsupervised outlier detection approach based on combining subspace clustering and multiple evidence accumulation to detect several kinds of intrusions [31]. They evaluate the method using KDDcup99 and two other real-life datasets.

7.2.2 Cluster Stability Analysis

Several cluster stability analysis techniques have been proposed in the literature [361–364]. We analyze cluster stability for identifying the actual number of clusters

generated by our clustering algorithm using stability calculation. Lange et al. introduce a cluster stability measure to validate clustering results [361] It determines the number of clusters by minimizing the classification risk of their measure. An experimental analysis of cluster stability measures for the identification of the number of clusters is discussed by [362]. Ben-David et al. provide a formal definition of cluster stability with specific properties [363]. They conclude that stability can be determined based on the behavior of the objective function. If the objective function is a unique global optimizer, the algorithm is stable. Das and Sil also present a cluster validation method for stable cluster generation using stability analysis [364].

7.2.3 Cluster Labelling

Cluster labelling is a challenging issue in unsupervised network anomaly detection. Most common cluster validity measures are summarized in [365–367]. Validity measures are usually based on internal and external properties of clustering results. Normally, internal validity measures obtain the compactness, connectedness and separation of the cluster partitions. External validity measures assess agreement between a new clustering solution and the reference clusters of interest [365] Jun [367] presents an ensemble method for cluster analysis It uses a simple voting mechanism for making decision from the results obtained by using several cluster validity measures Labelling of a cluster is must in case of cluster based unsupervised network anomaly detection Our proposed cluster labelling technique works based on the cluster size. compactness and the dominating feature set

7.2.4 Discussion

We provide a generic comparison of some published works on network anomaly detection [7, 29–31, 64, 164, 368] in Table 7.1. Based on a review of existing techniques for clustering based anomaly detection, cluster stability analysis and cluster labelling, we observe the following.

- Although many clustering based network intrusion detection techniques have been reported in the literature [7, 64, 164, 368], only a few have full features of an unsupervised intrusion detection system [7]. Many methods use only

clustering techniques for network anomaly detection without having cluster labelling strategies Hence there is still room to develop a full featured unsupervised network anomaly detection technique

- Existing stability analysis techniques have been mostly applied to analyze non-intrusion data But network traffic data is high dimensional and voluminous Thus there is scope for further enhancement in the network anomaly detection domain

- Only a very few labelling techniques are available in the literature [365–367] An appropriate use of indices can help in developing an effective labelling technique, which can support unsupervised anomaly detection to a great extent

Table 7.1: Comparison of unsupervised network anomaly detection methods

Author(s)	Method	Offline / Online	Packet /Flow level	Data Type	Unknown attack handled	Detection criteria	Full / Reduced space
Portnoy et al [64], 2001	Clustering-based	offline	packet	numeric	yes	cluster size distance	Full
Leung and Leckie [164], 2005	Clustering-based	offline	packet	numeric	no	distance, boundary value	Full
Jiang et al [7] 2006	Clustering-based	offline	packet	categorical	yes	distance	Full
Bhuyan et al [368], 2011	Outlier-based	offline	packet	numeric	yes	distance	Full
Song et al [29], 2011	Clustering-based	offline	packet	numeric	yes	distance	Full
Casas et al [31], 2012	Clustering-based UNIDS	offline	flow	numeric	yes	distance	Reduced
Noto et al [30], 2012	Model-based	offline	other	numeric	no	distance	Full

Due to these reasons, we see an opportunity to develop an integrated unsupervised network anomaly detection method

7.3 Problem Statement

Our work analyzes large amounts of network traffic data over an optimal and relevant feature space without any prior knowledge to identify anomalous or non-conforming test instance(s) with minimum false alarm The problem is defined as

follows. Let $\mathbb{X}$ be a collection of network traffic data with n data objects, where each object has $\mathbb{F}$ features The problem is to analyze $\mathbb{X}$ over an optimal and relevant feature subspace F', where $1 \leq F' \leq \mathbb{F}$ to identify groups of similar instances, C_i, where each C_i is labeled either as normal or anomalous.

The proposed method works in two phases (a) TreeCLUSS creates k clusters, i.e., $C_1, C_2, \cdots C_k$ from dataset $\mathbb{X}$ using a subset of relevant features, F', where each C_i is evaluated in terms of stability by using the function StableCLUSS, and (b) CLUSSLab labels each cluster, C_i based on the two assumptions: (i) The majority of network connections are normal, and (ii) Intra-similarity among the attack traffic instances is high. CLUSSLab exploits cluster size, compactness, dominating feature subset and outlier scores to label each cluster.

7.4 Unsupervised Network Anomaly Detection : The Framework

The main aim of this work is to detect network anomalies using an unsupervised approach with a minimum amount of false alarms. It can detect network anomalies without relying on existing signatures, training or labeled data. The proposed approach runs in two consecutive phases for analyzing network traffic in contiguous time slots of fixed length Figure 7.1 provides a conceptual framework of the proposed unsupervised network anomaly detection method.

In the *first* phase, we introduce a tree based subspace clustering technique (TreeCLUSS) for generating clusters in high dimensional large datasets. It is well known that network intrusion dataset is high dimensional and large. We apply our technique over a subset of features TreeCLUSS uses the MMIFS technique [81] for finding a highly relevant feature set. It uses a subset of features during cluster formation while not using any class labels. We analyze the stability of the cluster results obtained Cluster stability analysis for real life data is not a trivial task. It is performed using an ensemble of several index measures, viz., Dunn index [82], C-index (C) [84], Davies Bouldin index (DB) [83], Silhouette index (S) [87] and Xie-Beni index (XB) [93]. We choose a stable set of clusters when a certain number of clusters produces better result after multiple execution of this module.

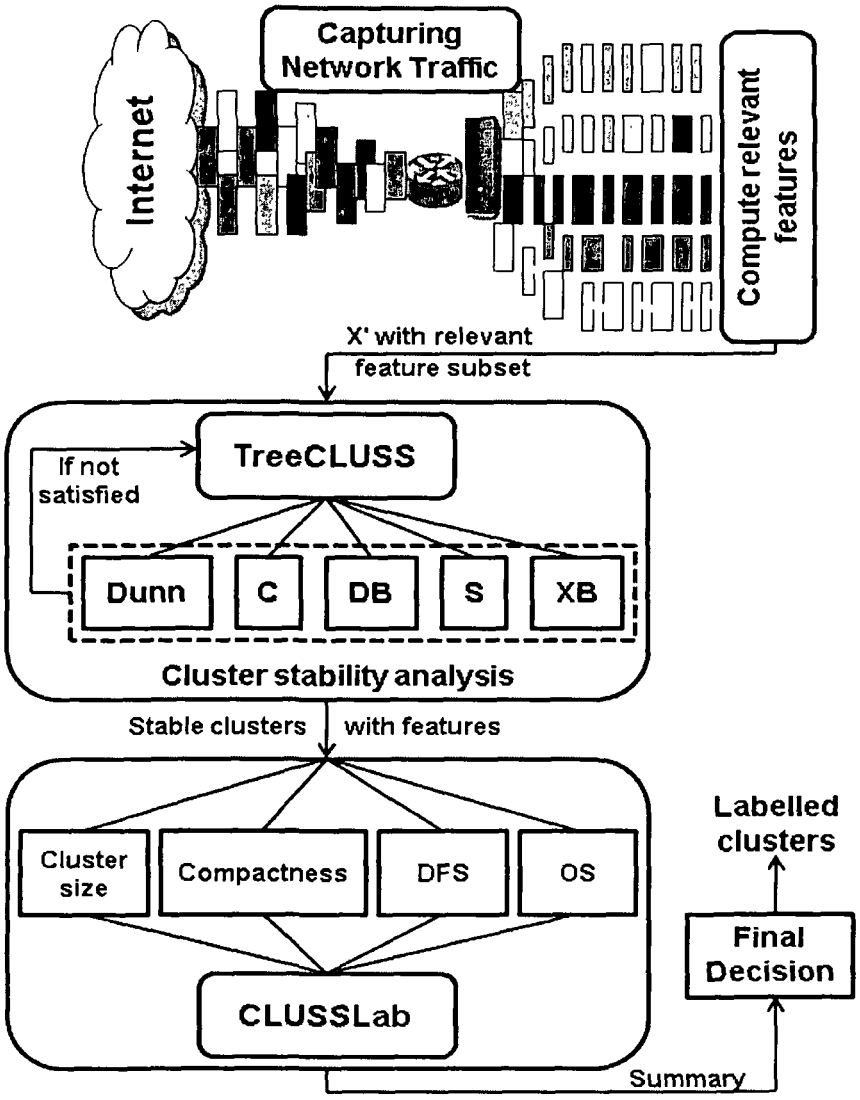


Figure 7.1: High level description of the unsupervised network anomaly detection method

In the *second* phase, we apply a cluster labelling technique (CLUSSLab) to label the stable clusters using a multi-objective approach. CLUSSLab takes into account the following features: cluster size, compactness obtained from the ensemble of five index measures, dominating feature subset (DFS) obtained for each cluster based on unsupervised feature clustering technique discussed in Section 7.4.3, and outlier score (OS) obtained based on the RODD technique [159]. Finally, we label each cluster as normal or anomalous based on the described measures.

7.4.1 TreeCLUSS: The Clustering Technique

TreeCLUSS is a tree based subspace clustering technique for high dimensional data It is especially tuned for unsupervised network anomaly detection It uses the MMIFS technique [81] to identify a subset of relevant features TreeCLUSS depends on two parameters viz , initial node formation threshold (α_2) and a step down ratio (ε) to extend the initial node depth-wise Both parameters are computed using a heuristic approach We now present notations, definitions and a lemma which help in the description of the TreeCLUSS algorithm

Definition 7.4.1. ***Data Stream*** - *A data stream $\mathbb{X}$ is denoted as $\{x_1, x_2, x_3 \cdots x_n\}$ with n objects, where x_i is the i^{th} object described with a d-dimensional feature set, i e , $x_i = \{f_{i1}, f_{i2}, f_{i3} \quad f_{id}\}$*

Definition 7.4.2. ***Neighbor of an object*** - *An object x_i is a neighbor of x_j over a subset of relevant features F', w r t a threshold α_2, iff $sim_f(x_i\ x_j) \leq \alpha_2$, where sim is a distance measure*

Definition 7.4.3. ***Connected objects*** - *If object i_i is a neighbor of object x_j and x_j is a neighbor of x_k w r t α_2, then x_i, x_j, x_k are connected*

Definition 7.4.4. ***Node*** - *A node N_i in the l^{th} level of a tree is a non-empty subset of objects x', where for any object $x_i \in N_i$, there must be another object $x_j \in x'$, which is a neighbor of x_i, and x_i is either (a) itself an initiator object or (b) is within the neighborhood of another initiator object $x_j \in N_i$*

Definition 7.4.5. ***Degree of a node*** - *The degree of a node N_j w r t α_2 is defined as the number of objects in N_j that are within α_2-neighborhood of any object $x_j \in N_j$*

Definition 7.4.6. $L_{1,i}^{F'\,\alpha_2}$ ***cluster*** - *It is a set of connected objects C_i at level 1 w r t α_2, where for any two objects $x_i, x_j \in C_i$ the neighbor condition (*Definition 7 4 2*) is true with reference to F_i'*

Definition 7.4.7. $L_{2,i}^{F'\,\beta_2}$ ***cluster*** - *It is a set of connected objects C_j at level 2 w r t β_2, where for any two objects $x_i, x_j \in C_j$ the neighbor condition (*Definition 7 4 2*) is true with reference to F_i' and $\beta_2 \leq (\frac{\alpha_2}{2} + \varepsilon)$ Also, $L_{2\,i}^{F'\,\beta_2} \subseteq L_{1,i}^{F'\,\alpha_2}$*

Definition 7.4.8. Outlier - *An object $x_i \in \mathbb{X}$ is an outlier if x_i is not connected with any other object $x_j \in \mathbb{X}$, where $x_j \in L_{1\,i}^{F'\,\alpha_2}$ In other words, x_i is an outlier if there is no $x_j \in \mathbb{X}$, so that x_i and x_j are neighbors (as per* Definition 7 4 2*)*

Lemma 7.4.9. *Two objects x_i and r_j belonging to two different nodes are not similar*

Proof Let $r_i \in N_i$ $r_j \in N_j$ and r_i is a neighbor of r_j According to *Definition 7 4 2* and *Definition 7 4 4*, x_i and x_j should belong to same node Therefore we come to a contradiction and hence the proof □

We present our TreeCLUSS algorithm for network anomaly detection in Algorithms 9 and 10 TreeCLUSS starts by creating a tree structure in a depth-first manner with an empty root node The root is at level 0 and is connected to all the nodes in level 1 The nodes in level 1 are created based on a maximal subset of relevant features by computing proximity within a neighborhood w r t an initial cluster formation threshold α_2 The tree is extended depth-first by forming lower level nodes w r t $(\frac{\alpha_2}{2}+\varepsilon)$ where ε is a controlling parameter of the step down factor i e , $\frac{\alpha_2}{2}$ α_2 and ε are computed using a heuristic approach A proximity measure *sim* is used in TreeCLUSS during cluster formation Although *sim* is free from the restriction of using a specific proximity measure, we used Euclidean distance to construct the tree from $\mathbb{X}$

The algorithm is illustrated using an example Let $\mathbb{X}$ be a dataset of d dimensions with details given in Table 7 2 Let $\mathbb{X} = \{r_1, r_2, \quad r_{16}\}$ and $\mathbb{F} = \{f_1, f_2 \quad f_{10}\}$ The extracted relevant feature set is given in Table 7 3 The class specific relevant features are identified from $\mathbb{X}$ w r t a threshold γ_2 We achieved best results when $\gamma_2 \geq 1$ for class C_1, $\gamma_2 \geq 0\ 918$ for class C_2 and $\gamma_2 \geq 0\ 917$ for class C_3 as shown in Table 7 3 An example tree obtained from $\mathbb{X}$ is shown in Figure 7 2 with reference to the reduced feature space as given in Table 7 3

7.4.2 Cluster Stability Analysis

We analyze the stability of clusters obtained from TreeCLUSS and several other clustering algorithms, viz k-means, fuzzy c-means, and hierarchical clustering A general stability comparison among these clustering algorithms w r t detection rate using the TUIDS datasets is given in Figure 7 3 The TUIDS datasets were built by us using our own testbed with different types of attacks (more details are given in 7 5 1) We propose an ensemble based cluster stability analysis technique based

Algorithm 9 Part 1 TreeCLUSS ($\mathbb{X}, \alpha_2\ \beta_2$)

Input: $\mathbb{X}$, the dataset α_2, threshold for L_1 cluster formation β_2, threshold for L_2 cluster formation
Output: set of clusters $C_1, C_2, C_3, \quad C_k$

```
1  initialization node_id ← 0                ▷ node_id is increased by 1 for new node
2  function BUILDTREE(𝕏, node_id)                                  ▷ function to build tree
3     for i ← 1 to 𝕏 do
4        if (x_i classified ! = 1 and check_ini_feat(MMIFS(x_i)) == true) and
   sim(x_i, x_j) ≤ α_2 then
5           CreateNode(x_i no, p_id, temp, node_count node_id, l)   ▷ function to
   create new node
6           while (F' - (l - 1)) ≥ θ do                  ▷ check relevant features subset
7              l++
8              for i ← 1 to 𝕏 do
9                 if x_i classified ! = 1 then ▷ if object is classified then labelled
   as 1
10                   p_id = check_parent(x_i no, l)
11                   if (p_id > -1 and check_ini_feat(MMIFS(x_i)) == true)
   then                                          ▷ function to check parent id
12                      CreateNode(x_i no, p_id, temp, node_count, node_id, l)
13                   end if
14                end if
15             end for
16          end while
17          l = 1
18       end if
19    end for
20 end function
21 function CREATENODE(no, p_id, temp, node_count id, l)   ▷ function to create
   node
22    node_id = new node()
23    node_id temp = temp
24    node_id nodel_count = node_count                    ▷ number of nodes in a level
25    node_id p_node = p_id
26    node_id id = id,
27    node_id level = l
28    ExpandNode(no, id, node_id temp, node_count l)      ▷ expand node in
   depth-wise for a particular node
29    temp = NULL
30    node_count = 0
31    node_id++
32 end function
33 function EXPANDNODE(no, id temp, node_count, l) ▷ function to expand node
34    if X_no classified == 1 then
35       return
36    else
```

Algorithm 10 Part 2 TreeCLUSS ($\mathbb{X}, \alpha_2\ \beta_2$)

```
37        X_no classified = 1
38        X_no node_id = id
39        for i ← 1 to 𝕏 do
40            if (x_i classified ! = 1) then
41                minRank_F' = find_minRank(MMIFS(x_i)) ▷ select next subset of relevant features
42                if (F' - minRank_F') ≥ 0 then ▷ check maximum height of the tree
43                    minRank_F'++ until get a specific cluster otherwise stop ▷ continue for getting specific class
44                    ExpandNode(x_i no, id, temp, temp_count, 1) ▷ expand node in depth-wise
45                end if
46            end if
47        end for
48    end if
49 end function
50 function StableCLUSS(C_k)                ▷ function to analyze cluster stability
51    for i ← 1 to k do
52        for j ← 1 to 5 do
53            VI_c[j] = compute(I_ci)▷ compute validity index and store values into an array
54            if (VI_c[j] ≥ σ1 or VI_c[j] ≤ σ2) then      ▷ check threshold for each validity index
55                VI_c[j] = 1
56            else
57                VI_c[j] =0
58            end if
59        end for
60        if (C_i = Max(VI_c[i])) then ▷ check for maximum validity index value for cluster stability
61            stable cluster C_i
62            Return Max(VI_c[i])
63        else
64            go to step 2
65        end if
66    end for
67 end function
```

on Dunn index [82] C-index (C) [84] Davies Bouldin index (DB) [83], Silhouette index (S) [87] and Xie-Beni index (XB) [93] (shown in Figure 7 1) We choose several well known cluster validity measures for stability analysis We analyze each cluster based on distance to reduce computational overhead All our measures are distance based We briefly discuss each measure along with the values expected for

Table 7.2: Sample dataset X and CL in the last column is the class label

Object ID	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	*CL*
O_1	9 23	0 71	2 43	0 60	104	2 80	3 06	0 28	2 29	5 64	1
O_2	22 53	6 51	6 64	4 96	72 79	2 60	11 63	0 80	9 00	36 97	8
O_3	16 37	5 76	1 16	11 88	95	5 50	1 10	0 49	6 87	20 45	5
O_4	10 37	1 95	9 50	0 80	110	1 85	2 49	0 64	3 18	9 80	2
O_5	14 67	4 85	1 92	11 94	96	4 10	1 79	0 12	4 73	10 80	4
O_6	9 20	0 78	2 14	0 20	103	2 65	2 96	0 26	2 28	4 38	1
O_7	12 37	0 84	1 36	11 60	95	3 98	1 57	0 98	1 42	10 95	3
O_8	9 16	1 36	9 67	0 60	110	1 80	2 24	0 60	3 81	9 68	2
O_9	16 17	5 86	1 53	11 87	93	5 89	1 75	0 45	6 73	20 95	5
O_{10}	18 81	6 31	4 40	4	70	2 15	8 09	0 57	7 83	27 70	6
O_{11}	14 64	4 82	1 02	11 80	94	4 02	1 41	0 13	4 62	10 75	4
O_{12}	20 51	6 24	5 25	4 50	70 23	2	9 58	0 60	8 25	32 45	7
O_{13}	12 33	0 71	1 28	11 89	96	3 05	1 09	0 93	1 41	10 27	3
O_{14}	20 60	6 46	5 20	4 50	71	2 42	9 66	0 63	8 94	32 10	7
O_{15}	18 70	6 55	5 36	4 50	73 24	2 70	8 20	0 57	7 84	27 10	6
O_{16}	22 25	6 72	6 54	4 89	69 38	2 47	10 53	0 80	9 85	36 89	8

Table 7.3: Relevant feature set, F' and attribute rank values

Class	*Object ID*	*Relevant feature set*	*Feature rank value*
C_1	O_1,O_4 O_6,O_8	f_5,f_6,f_2,f_3 f_9,f_{10},f_7,f_8	1,1,1,1,1,1,1,1
C_{11}	O_1,O_6	f_5,f_6,f_2,f_3 f_9,f_{10},f_7	1,1,1,1,1,1,1
C_{12}	O_4,O_8	f_5,f_6,f_2,f_3 f_9,f_{10}	1,1,1,1,1,1
C_2	O_3,O_5 O_7,O_9,O_{11},O_{13}	f_1, f_2, f_6, f_9, f_8 f_{10}	1 585 1 585,1 585,1 585, 1 585 0 918
C_{21}	O_3, O_9	f_1, f_2, f_6, f_9, f_8	1 585 1 585,1 585,1 585,1 585
C_{22}	O_5, O_{11}	f_1, f_2, f_6, f_9	1 585 1 585,1 585,1 585
C_{23}	O_7, O_{13}	f_1, f_2, f_6, f_8	1 585 1 585,1 585,1 585
C_3	O_2,O_{10},O_{12} O_{14} O_{15} O_{16}	f_7 f_1, f_{10}, f_8 f_9 f_4, f_3	1 584 1 584,1 584,1 584,1 584,0 917 0 917
C_{31}	O_2, O_{16}	f_7, f_1, f_{10}, f_8 f_9 f_4	1 584 1 584,1 584,1 584,1 584,0 917
C_{32}	O_{10}, O_{15}	f_7, f_1, f_{10}, f_8 f_9	1 584 1 584,1 584,1 584,1 584
C_{33}	O_{12}, O_{14}	f_7, f_1, f_{10}, f_8 f_4	1 584 1 584,1 584,1 584, 0 917

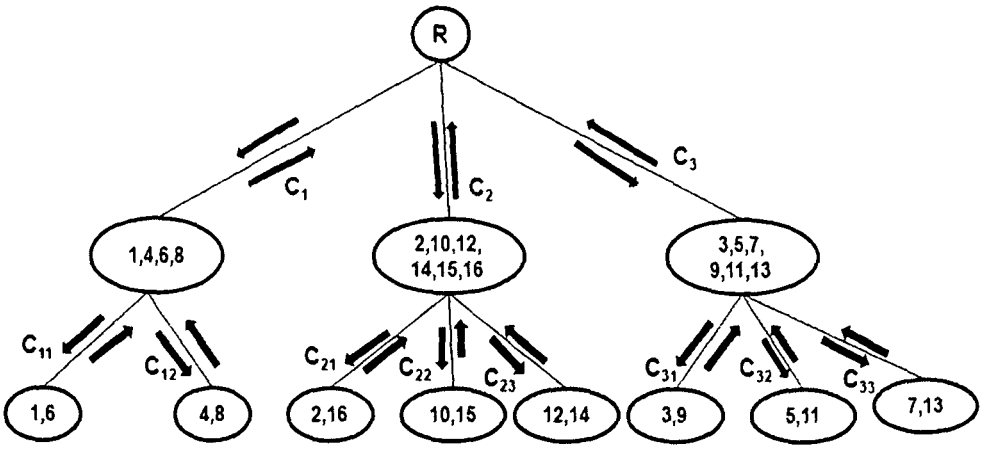


Figure 7.2: Tree obtained from X, given in Table 7 2

good clusters in Table 7 4

We pass each cluster C_i to a function StableCLUSS to measure stability It computes all the indices for each of the clusters C_1, C_2 $, C_k$ If it judges that the result is good for an index, it stores a 1, otherwise assigns 0 It computes 1 or

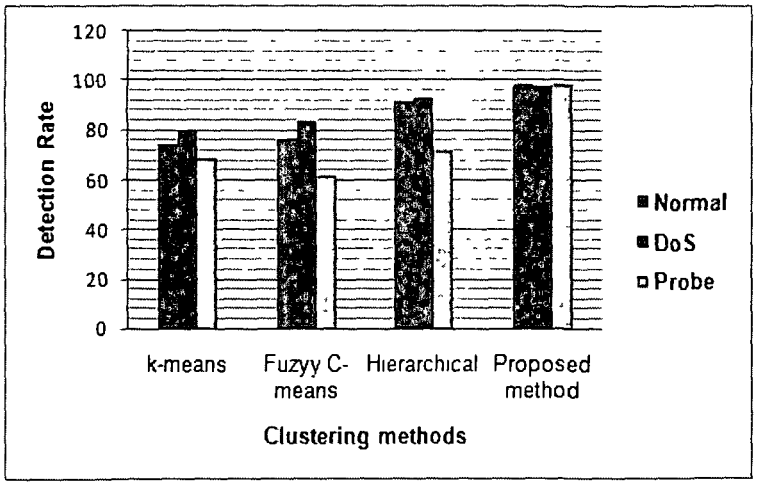


Figure 7.3: Comparison of stability analysis with various algorithms using TUIDS packet level intrusion dataset

0 for each of the indices as given below. σ_1 and σ_2 are threshold parameters.

$$V_i = \begin{cases} 1, & I_i \geq \sigma_1 \ \ or \ \ I_i \leq \sigma_2 \\ 0, & \text{otherwise} \end{cases}$$

Finally, we take the maximum number of occurrences of 1 to decide if a cluster is stable or not. If a cluster C_i is not stable, it sends control back to TreeCLUSS to regenerate another set with a different number of clusters. We choose the best set of stable clusters after we execute the module multiple times.

7.4.3 CLUSSLab : The Cluster Labelling Technique

CLUSSLab is a multi-objective cluster labelling technique for labelling the clusters generated by TreeCLUSS. It decides the label of the instances of a cluster based on a combination of the following measures: (a) cluster size, (b) compactness, (c) dominating feature subset and (d) outlier score of each instance. Each measure is described next.

(a) *Cluster size*: It is the number of instances in a cluster.

(b) *Compactness*: To find the compactness of a cluster C_i, obtained from TreeCLUSS, we use the five very well known indices as given in Table 7.4 and discussed earlier in Section 7.2.2.

Table 7.4: Cluster stability measures : definition. features and criteria for better cluster

Stability measures	*Definition*	*Features*
Dunn index (Dunn) [82]	$\frac{d_{min}}{d_{max}}$ where d_{min} denotes the smallest distance between two objects from different clusters and d_{max} is the largest distance between two elements within the same cluster.	(a) Computed for finding compact and well separated clusters. (b) Larger values of *Dunn* indicate better clustering i.e., the range is $(0, \infty)$.
C-index (C) [84]	$\frac{S-S_{min}}{S_{max}-S_{min}}$, 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 sum of n smallest distances and n largest distances, respectively.	(a) Used to find cluster quality when the clusters are similar sizes. (b) Smaller values of C indicate better clusters, i.e., the range is $(0, 1)$.
Davies Bouldin index (DB) [83]	$\frac{1}{n}\sum_{i=1, i\neq j}^{n} max(\frac{\sigma_j+\sigma_j}{d(c_i\ c_j)})$, where n is the number of clusters· σ_i is the average distance of all patterns in cluster i to their cluster center. c_i; σ_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 between the cluster centers c_i and c_j.	(a) Lower value of DB indicates better clusters, i.e., the range is $(0, \infty)$. (b) It has low computational cost and can find better clusters of spherical shape.
Silhouette index (S) [87]	$\frac{b_i-a_i}{max\{a_i,b_i\}}$. where a_i is the average dissimilarity of 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.	(a) Computed for a cluster to identify tightly separated groups. (b) Better if the index value is near 1, i.e., the range is $(-1. 1)$.
Xie Beni index (XB) [93]	$\frac{\pi}{N\, d_{min}}$, where $\pi = \frac{\sigma_i}{n_i}$. is called compactness of cluster i. Since n_i is the number of points in cluster i. σ is the average variation in cluster i; $d_{min} = min\|\|k_i - k_j\|\|$.	Smaller values of XB are expected for compact and well-separated clusters, i.e.. the range is $(0, 1)$.

(c) *Dominating feature subset* The subset of features which mostly influences the formation of the clusters is referred to as a dominating feature subset. We identify the dominating features using an adaptive unsupervised feature clustering technique (UReFT) based on Renyi's entropy [369]. Renyi's entropy performs non-parametric estimation by avoiding the problems of the traditional entropy metric. Renyi's entropy with probability density function (pdf) f_x for a stochastic variable x and Renyi's constant λ is given by

$$H_R(x) = \frac{1}{1-\lambda} ln \int f_x^{\lambda} dx, \lambda > 0, \lambda \neq 1 \tag{7.1}$$

Renyi's quadratic entropy is defined by [370] when $\lambda = 2$ as follows, assuming a Gaussian pdf:

$$\begin{aligned} H_R(x) &= -ln \int f_x^2 dx \\ &= -ln \left(\frac{1}{N} \sum_{i=1}^{N} G(x - x_i, \sigma^2) \right) \left(\frac{1}{N} \sum_{i=1}^{N} G(x - x_j, \sigma^2) \right) \\ &= -ln \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} G(x_i - x_j, 2\sigma^2) \end{aligned} \tag{7.2}$$

where G is the Gaussian kernel, σ is the smoothing parameter (we found better results when $\sigma = 0.9$ to 0.12), x_i and x_j are the i^{th} and j^{th} features of N data objects. We also note that

$$G(x_i - x_j, 2\sigma^2) = \frac{1}{(2\pi)^{\frac{d}{2}} \sqrt{2\sigma^2}} exp \left(-\frac{(x_i - x_j)^2}{4\sigma^2} \right) \tag{7.3}$$

where d is the dimension of variable x. Assume that we obtain k feature clusters, i.e., $C = \{C_1, C_2, \cdots C_k\}$. A feature object x is assigned to a cluster C_i iff,

$$(H(C_i + x) - H(C_i)) < (H(C_k + x) - H(C_k)), k \neq i \tag{7.4}$$

where $H(C_k)$ denotes the entropy of cluster C_k. This method is referred to as differential entropy clustering [371]. We compute $H(C_k)$ and $H(C_i, C_j)$ for within and between cluster entropy as follows.

$$H(C_k) = -ln \frac{1}{N_k^2} \sum_{i=1}^{N_k} \sum_{j=1}^{N_k} G(x_i - x_j, 2\sigma^2) \tag{7.5}$$

$$H(C_i, C_j) = -ln \frac{1}{N_i N_j} \sum_{p=1}^{N_i} \sum_{q=1}^{N_j} G(x_p - x_q, 2\sigma^2) \tag{7.6}$$

The main goal of our technique is to identify a dominating feature set with the least redundancy and the most relevancy. Initially, we assume that each

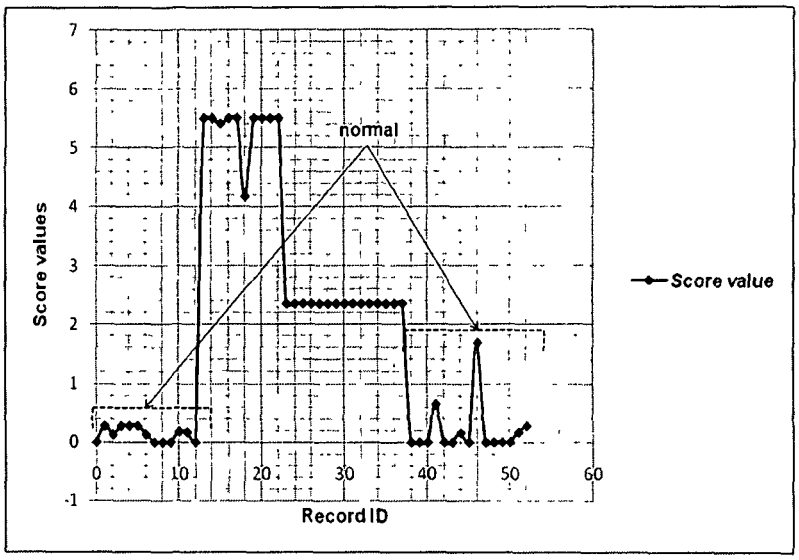


Figure 7.4: Identification of normal ranges using outlier score ranking over intrusion datatset

cluster contains two feature subsets (i) the selected or relevant subset and (ii) the non-selected or irrelevant subset. The selected cluster is the dominating features set and the non-selected cluster is the irrelevant feature set. The method starts with a single feature object, C_s and assigns another object to it by computing Renyi's entropy (using Equations 7.4, 7.5 and 7.6) w.r.t. a threshold η_1, otherwise it creates a new cluster, C_{ns} known as the non-selected cluster. It adaptively assigns each candidate feature object to C_s or C_{ns} w.r.t. threshold η_1 and the threshold for intra-cluster entropy η_2. The threshold values of η_1 and η_2 are also chosen based on a heuristic approach.

(d) *Outlier score*: Here, we exploit our own outlier identification algorithm, RODD [159] to compute the score of each instance with reference to the normal profiles. A graph is plotted based on sorted outlier ranking against those instances as shown in Figure 7.4 and from the graph, a cutoff is decided to distinguish the normal from anomalous instances. We see in the graph that for any two-class combination such as (normal, DoS), (normal, probe), (normal,U2R), or (normal, R2L) with various proportions, it is still possible to distinguish the normal from the rest.

Based on the cluster size, compactness, dominating features identified using UReFT and interval of outlier score rank values, we label each cluster as anomalous or normal w r t the thresholds We found best result for labelling each cluster as anomalous with matching probability, $P' \leq 0.63$ w r t the above measures The CLUSSLab algorithm is given as Algorithm 11 It is a multi-objective technique to label each cluster as normal or anomalous UReFT is the unsupervised Renyi's entropy based feature clustering technique to identify the relevant features set for each cluster It matches the existing class specific feature set while labelling

Algorithm 11 CLUSSLab($C_k, \xi_1\ \xi_2, \xi_3, \xi_4$)

Input: C_k represents the cluster obtained from TreeCLUSS ξ_1 is number of instances in a cluster ξ_2 is the cluster compactness score ξ_3 is the matching probability of features of a cluster with respect to a specific class and ξ_4 is the outlier score value of each instance of a cluster

Output: Label clusters $C_1\ C_2, C_3 \quad C_k$ as normal or anomalous

1 **for** $i \leftarrow 1$ to k **do**
2 $S[i] = |C_i|$ ▷ S stores the cardinality of each cluster
3 M[i] = call StableCLUSS(C_i) ▷ M stores the cluster compactness score
4 **end for**
5 **function** UReFT(C_k) ▷ function to unsupervised feature clustering technique
6 **for** $i \leftarrow 1$ to k **do**
7 **for** $j \leftarrow 1$ to S_i **do**
8 **if** $(H(C_s)) \leq \eta_1$ && $(H(C_s\ C_{ns})) \leq \eta_2$ **then** ▷ check within cluster and between cluster entropy
9 $C_s[z] \leftarrow f_z,\ z = 1, 2, \quad d$
10 **else**
11 $C_{ns}[z] \leftarrow f_z,\ z = 1, 2, \quad d$
12 **end if**
13 **end for**
14 **end for**
15 **end function**
16 **for** $i \leftarrow 1$ to k **do**
17 **if** $S[i] \leq \xi_1$ && $M[i] < \xi_2$ && $C_i \geq \xi_4$ **then** ▷ check cardinality of a cluster, compactness score, and outlier score
18 **if** $P'(|C_s[z]|, |MMIFS[z]|) \leq \xi_3$ **then** ▷ check matching probability w r t a specific class
19 *anomalous* $\leftarrow C_i$
20 **else**
21 *normal* $\leftarrow C_i$
22 **end if**
23 **end if**
24 **end for**

7.4.4 Complexity Analysis

As discussed. the proposed method ıs works in two phases. The first phase ıs subspace clustering technique, i.e., the TreeCLUSS. We assume that k clusters arc obtaincd from n data objects. During cluster formation, TrceCLUSS takcs $O(nlogk)$ time and for stability analysis, it takes $O(klogk)$ time. Hence, the total computational complexity of TreeCLUSS is $O(nlogk)$.

The second phase is multi-objective cluster labellıng technique, i.e., the CLUS-SLab. It is again comprised of four sub-modules viz., cluster size, compactness, dominating feature subset (DFS) and outlier score (OS) To compute, compactness, dominating fcaturc subsct and outlier scorc, it takcs $O(nlogn)$, $O(n)$, and $O(kn)$ time, respectively. Hence, the total time complexity of CLUSSLab is $O(nlogn + kn)$

Thc time complcxity for cach stagc of our unsupcrviscd nctwork anomaly detcc-tion method is linear w.r.t. the size of dataset, the number of features, the number of clusters and the labelling of each clusters. Hence, it is effective in detecting known as wcll as unknown attacks with thc lcast amount of falsc alarms.

7.5 Experimental Analysis

In this section, we present experimental analysis and results of the unsupervised network anomaly detection method usıng several real world datasets from the UCI machinc lcarning rcpository and datascts prepared in thc TUIDS tcstbcd at both packet and flow levels [264]. The datasets used in this work to evaluate the proposed method and experimental results are discussed below.

7.5.1 Datasets Used

We use two sets of datasets, viz., (a) Non-intrusion datasets taken from UCI ML rcpository for initial evaluation and cstablishmcnt of the proposed algorithms, (b) intrusion datascts.

Non-intrusion Datasets

We use ten non-intrusion datasets [356]. Zoo, Glass, Abalone, Shuttle, Wine, Lymphography, Heart, Pima, Vehicle and Poker Hand to initially validate clusters generated by TreeCLUSS. Table 7 5 describes the details of the non-intrusion datasets and their characteristics.

Table 7.5: Characteristics of real-life non-intrusion datasets

Non-intrusion Datasets (NID)	Datasets	Dimension	No of instances	No of classes
NID1	Zoo	18	101	7
NID2	Glass	10	214	6
NID3	Abalone	8	4177	29
NID4	Shuttle	9	14500	3
NID5	Wine	13	178	3
NID6	Lymphography	18	148	4
NID7	Heart	13	270	2
NID8	Pima	8	768	2
NID9	Vehicle	18	846	4
NID10	Poker Hand	10	25010	10

Intrusion Datasets

We use five different real life intrusion datasets. such as (a) TUIDS coordinated scan datasets, (b) TUIDS datasets, (c) TUIDS DDoS datasets, (d) NSL-KDD dataset and (e) KDDcup99 datasets. We capture, preprocess, and extract features in both packet and flow level network traffic and generate TUIDS benchmark datasets. A detailed discussion of TUIDS datasets generation is given in Chapter 4 Next we discuss each dataset in brief.

(a) *TUIDS real-time Coordinated scan dataset*: We launched attacks in a coordinated mode using the *nmap*[1] tool to generate the traffic including normal traffic. We captured the traffic in both packet and flow levels to prepare the dataset. Characteristics of this dataset are given in Table 7.6.

(b) *TUIDS real-life intrusion dataset*: This dataset is prepared by launching 20 different attacks with normal traffic connections It contains 15 DoS attacks and 5 probe attacks Characteristics of this datasets are given in Table 7.6.

[1]http //nmap.sourceforge.net/

(c) *TUIDS real-life DDoS dataset* It is prepared using the same TUIDS testbed with three different flooding attacks launched in amplification mode while capturing the traffic at flow level only Characteristics of this dataset are given in Table 7 6 A brief description of DDoS attacks we launched is given below

- In *smurf* attack, the attacker sends packets to a network amplifier (a system supporting broadcast addressing) with the return address spoofed to the victim's IP address It uses ICMP ECHO packets and as a result the original packet spoofs tens or even hundreds of times to the victim host
- The *Fraggle* attack is similar to a smurf attack in that the attacker sends packets to a network amplifier but uses UDP ECHO packets instead of ICMP ECHO packets The UDP ECHO packets are sent to the port that supports character generation (chargen, port 19 in Unix systems), with the return address spoofed to the victim's echo service (echo, port 7 in Unix systems) creating an infinite loop
- The *SYN flooding* attack exploits the TCP s three-way handshake mechanism and its limitation in maintaining half-open connections So it drops more packets while sending from source to destination

(d) *NSL-KDD intrusion datasets* NSL-KDD[1] is an enhanced version of the KDDcup99 datasets These are well-known datasets for intrusion detection system evaluation The dataset is described in Table 7 6

(e) *KDDcup99 intrusion datasets* This is the most well-known and the most popular intrusion dataset used for evaluation of any intrusion detection system It contains training data processed into about five million network connection records A connection record is a sequence of TCP packets with well-defined starting and ending times Each connection record is unique in the dataset with 41 continuous and nominal features plus one class label The features available in the KDDcup99 dataset are reported in Chapter 4 A detailed description of the dataset is also given in Table 7 6

[1]http //www iscx ca/NSL-KDD/

Table 7.6: Distribution of Normal and Attack connection instances in real-life TUIDS Coordinated scan (packet and flow) TUIDS (packet and flow) TUIDS DDoS flow level, NSL-KDD packet level and KDDcup99 packet level intrusion datasets

Intrusion Datasets (ID)	Connection type	Dimensions	No of instances	No of classes
ID1	*TUIDS coordinated scan packet level*			
	Normal		106380	1
	Probe	50	14423	6
	Total		120803	7
ID2	*TUIDS coordinated scan flow level*			
	Normal		36033	1
	Probe	25	15654	6
	Total		51687	7
ID3	*TUIDS packet level*			
	Normal		47895	1
	DoS	50	30613	15
	Probe		7757	5
	Total		86265	21
ID4	*TUIDS flow level*			
	Normal		16770	1
	DoS	25	14475	15
	Probe		9480	5
	Total		40725	21
ID5	*TUIDS DDoS flow level*			
	Normal	25	43252	1
	Flooding attacks		22707	3
	Total		65959	4
ID6	*NSL-KDD packet level*			
	Normal		9711	1
	DoS		7460	11
	Probe	41	2421	6
	R2L		2753	12
	U2R		199	8
	Total		22544	38
ID7	*KDDcup99 corrected packet level*			
	Normal		60593	1
	DoS		229853	12
	Probe	41	4166	6
	R2L		16189	12
	U2R		228	6
	Total		311029	37

7.5.2 Results and Discussions

In this section we report the performance of the proposed method using real-life and benchmark datasets The method does not use any class information when it processes a dataset for anomaly detection We measure the accuracy of the algorithms using the following metrics

- Detection rate = True Positive/(True Positive + False Negative)
- False positive rate = False Positive/(False Positive + True Negative)

Non-intrusion Datasets

The method was initially tested using non-intrusion datasets We label each cluster obtained by TreeCLUSS using our CLUSSLab cluster labelling technique. We compare performance in terms of detection rate (DR) and false positive rate (FPR). Detailed results are given in Table 7 7.

Table 7.7: Experimental results on non-intrusion datasets

Dataset	No. of clusters	Correctly detected	Misdetected	Detection rate (%)	False positive rate (%)
NID1	8	95	6	94.06	0.0594
NID2	9	206	8	96.26	0.0373
NID3	22	4002	175	95.81	0.0418
NID4	3	14296	204	98.59	0.0141
NID5	3	174	4	97.75	0.0121
NID6	5	135	13	91.22	0.0471
NID7	2	266	4	98.51	0.0522
NID8	2	761	7	99.08	0.0125
NID9	5	809	36	95.62	0.0613
NID10	12	24867	143	99.42	0.0018

Intrusion Datasets

In these experiments, we test our method for network anomaly detection using TUIDS, NSL-KDD and KDDcup99 network intrusion datasets discussed above. It converts all categorical attributes into numeric form and then computes $log_b(x_{ij})$ to normalize larger attribute values, where x_{ij} is a large attribute value and b depends on the attribute values Nominal features such as protocol (e.g., *tcp, udp, icmp*), service type (e.g., *http, ftp, telnet*) and TCP status flags (e.g., *sf, rej*) are converted into numeric features. We replace categorical values by numeric values. For example, in the protocol attribute, the value TCP is changed to 1, UDP is changed to 2 and ICMP is changed to 3

We initially apply TreeCLUSS on a subset of relevant features extracted using the MMIFS algorithm [81] for all intrusion datasets to generate a stable number of clusters and label each cluster using CLUSSLab as normal or anomalous. Experiments used the following datasets. (a) TUIDS real-time Coordinated scan datasets, (b) TUIDS real-time intrusion datasets, (c) TUIDS real-time DDoS datasets, (d)

NSL-KDD intrusion datasets, and (e) KDDcup99 intrusion datasets Then, we apply MMIFS algorithm to find the class specific relevant subspaces for all datasets These class specific feature subsets are used during cluster formation A list of relevant features for all datasets with their ranks in descending order are given in Table 7 8 Finally, experimental results of all datasets are given in Table 7 9

Table 7.8: Feature ranks for all classes in intrusion datasets See Table 7 6 for ID numbers

Datasets	#Features	Selected features
ID1		*packet level*
Normal	10	8 33 7 9,14,28,45,1,48 2
Probe	15	45 8 34,33,49,7,14 50,44,41,39,20 2 22,30
ID2		*flow level*
Normal	11	14 7 18,15,19,2,22 21,25,1,4
Probe	14	7,14 11,9,25,21,24 18,15,2,6,1,12,13
ID3		*packet level*
Normal	9	8,33 7 9,14,28,45,1,48 2
DoS	10	8,33 7 40,38,9,2,41,49 2
Probe	13	45 8 34,33,49,7,50 44,41,39,20,2,30
ID4		*flow level*
Normal	11	14 7 18,15,19,16 2 22,21,25,1
DoS	10	14 18,7,24,25,2,12 16,19,22
Probe	13	7,14 11,9,16,25,21 24,18,15,2,6,1
ID5		*flow level*
Normal	9	8 33 7 9,14,28,45,1,48
Flooding attacks	12	8 9 31 14,33,43,49 47,7,42,1,11
ID6		*packet level*
Normal	7	5 3 23 6,35,1,29
DoS	10	5,23 6 24,2,24,36,41 3 25
Probe	15	40 5 23,33,4,28,3,41 35,29,27,32,6 12 24
U2R	10	5 1 3 33,24,23,14,6,32 21
R2L	14	3,6 5 13,22,23,10,35 37,24,4,1,39,38
ID7		*packet level*
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, 22, 38 10, 3

Discussions

We achieve better results than competing algorithms for network anomaly detection in terms of detection rate and false positive rate A comparison of our method with several competing algorithms viz , C4 5 [372], ID3 [156], CN2 [373], CBUID [7], TANN [8], HC-SVM [4] using TUIDS datasets and KDDcup99 datasets is given in Figure 7 5 and Figure 7 6, respectively It can be easily seen from the figures that our method outperforms the competing algorithms [4,6–8] in the terms of detection rate and false positive rate, especially in case of probe, U2R and R2L attacks

TreeCLUSS depends on two main parameters, α_2 and β_2 but users need to

Table 7.9: Results on intrusion datasets using proposed method

Type of traffic	No. of clusters	Correctly detected	Mis-detected	Detection rate (%)	False positive rate (%)
ID1	*packet*	*level*			
Normal.	7	105121	1259	98.81	0.0164
Probe.	7	14292	131	99.09	0.0017
Overall.	14	119413	1390	98.95	0.0091
ID2	*flow*	*level*			
Normal.	5	35668	365	98.99	0.0153
Probe.	7	15519	135	99.13	0.0015
Overall.	12	51187	500	99.06	0.0084
ID3	*packet*	*level*			
Normal.	5	47109	786	98.35	0.0164
DoS.	16	29997	616	97.99	0.0166
Probe.	5	7637	120	98.45	0.0014
Overall.	26	84743	1522	98.26	0.0114
ID4	*flow*	*level*			
Normal.	3	16486	284	98.30	0.0169
DoS.	16	14381	101	99.35	0.0167
Probe.	4	9225	255	97.31	0.0149
Overall.	23	40092	640	98.32	0.0161
ID5	*flow*	*level*			
Normal.	2	43104	148	99.65	0.0034
Flooding attacks.	4	22272	435	98.08	0.0195
Overall.	6	65376	583	99.11	0.0089
ID6	*packet*	*level*			
Normal.	3	9573	138	98.57	0.0147
DoS.	12	7391	69	99.08	0.0052
Probe.	6	2356	65	97.32	0.0182
R2L.	11	2367	386	85.97	0.1493
U2R.	7	131	68	65.83	0.2050
Overall.	39	21818	726	89.35	0.0784
ID7	*packet*	*level*			
Normal.	5	59901	692	98.85	0.0113
DoS.	14	229796	57	99.97	0.0016
Probe.	5	4018	148	96.45	0.0160
R2L.	13	14007	2182	86.52	0.1335
U2R.	5	151	77	66.23	0.1973
Overall.	42	307873	3156	98.98	0.0102

provide α_2 value only. β_2 can be derived from α_2. Each is chosen using a heuristic approach for each dataset. Hence, our method is less dependent on input parameters compared to competing algorithms [4, 6–8, 31].

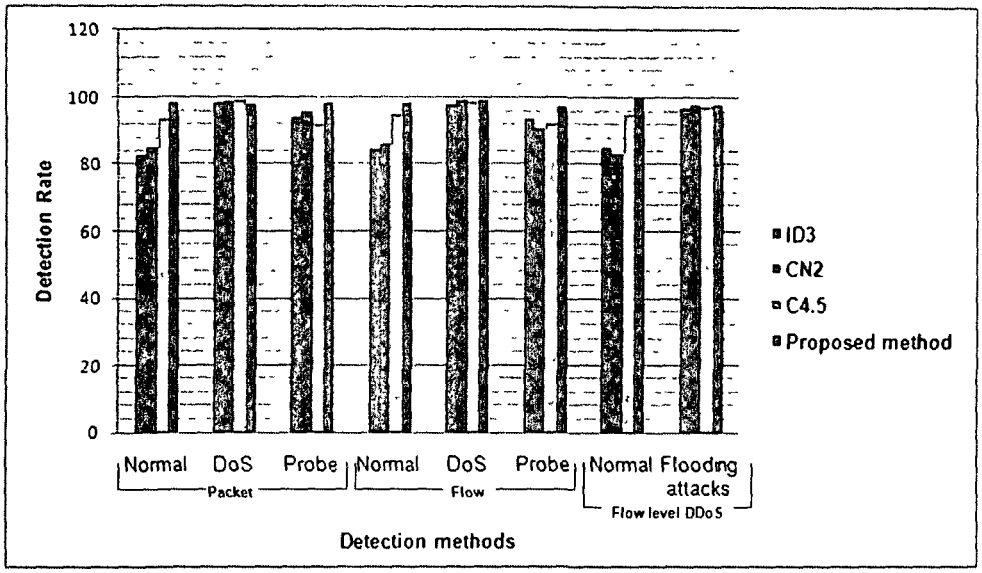


Figure 7.5: Comparison of our method with competing algorithms using TUIDS intrusion datasets

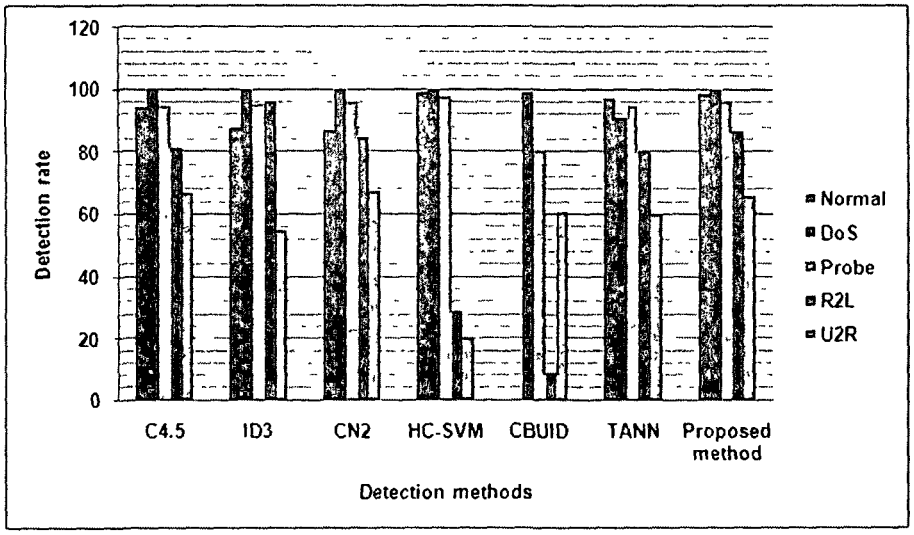


Figure 7.6: Comparison of proposed method with competing algorithms using KDD-cup99 intrusion datasets

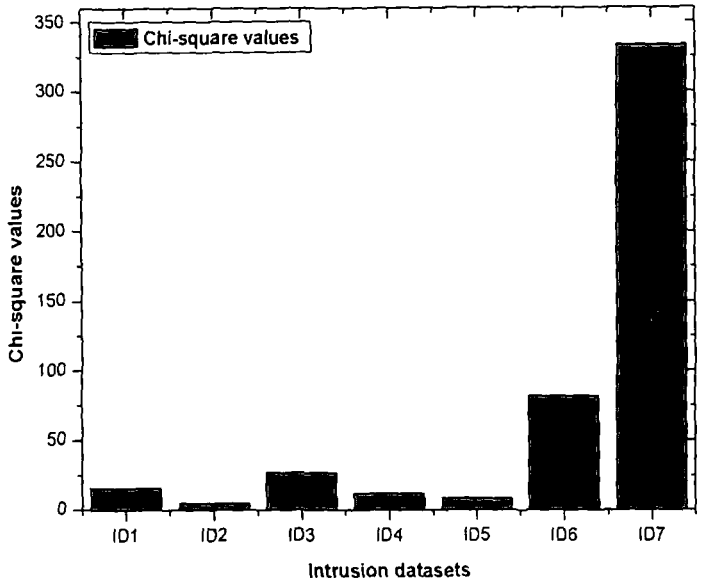


Figure 7.7: Chi-square test statistics for seven different intrusion datasets with significance level $\alpha = 0.05$ (min = 4.86, max = 333.28)

7.5.3 Statistical Significance Test

In addition to the evaluation based on real-life intrusion data, we also compute statistical significance of our results using two well known statistical measures: chi-square test and t-test. The chi-square test is used to compute how significantly the observed values are different from the expected values of the distribution for a given sample [374]. We reject the null hypothesis if the chi-square value is greater than the tabulated value w.r.t. the degree of freedom and level of significance. We tested over seven network intrusion datasets mentioned above and obtained significance level $\alpha = 0.05$ in all datasets as shown Figure 7.7.

The *t*-test is used to find the difference between two means in relation to the variation in the data. If the computed t-value exceeds the tabulated value, we say that it is highly significant, so that we can reject the null hypothesis. We tested over seven intrusion datasets and obtained t-values as shown in Figure 7.8. Thus, for both statistical significance tests, we achieved higher significance level for differences between normal and anomalous samples.

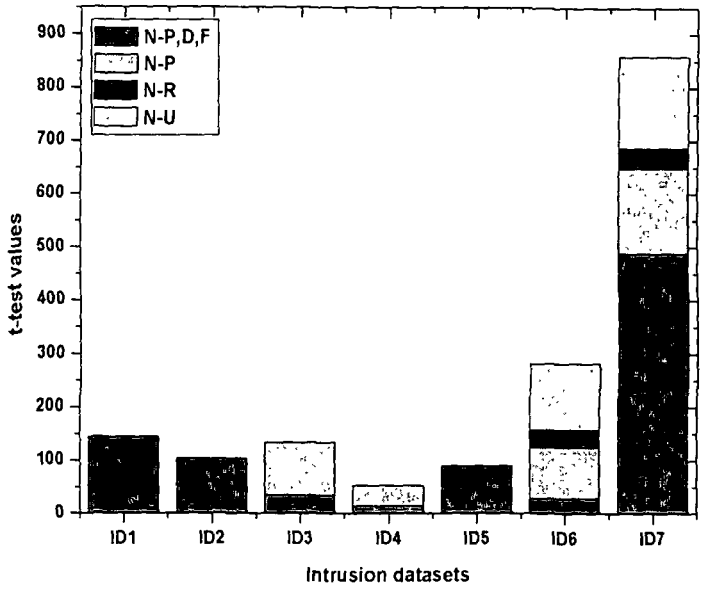


Figure 7.8: t-test statistics for seven different intrusion datasets with significance level $\alpha = 0.05$; N-P,D,F,R,U represents the normal, probe, DoS, flooding attacks, R2L and U2R respectively.

7.6 Summary

This chapter presents an unsupervised tree-based subspace clustering technique for network anomaly detection in high dimensional datasets. It generates the approximate number of clusters without having any prior knowledge of the domain. We analyze cluster stability for each cluster using an ensemble of multiple cluster indices. We also introduce a multi-objective cluster labelling technique to label each stable cluster as normal or anomalous. The major attractions of our proposed method are: (i) TreeCLUSS does not require the number of clusters apriori, (ii) It is free from the restriction of using a specific proximity measure, (iii) CLUSSLab is a multi-objective cluster labelling technique including an effective unsupervised feature clustering technique for identifying a dominant feature subset for each cluster, and (iv) TreeCLUSS exhibits a high detection rate and a low false positive rate, especially in case of probe, U2R, and R2L attacks. Thus, we are able to establish the proposed method to be superior compared to competing network anomaly detection techniques. We also demonstrate that the results produced by our method are statistically significant.

Chapter 8

Extended Entropy Metric-based Approach for DDoS Flooding Attack Detection

In the previous chapters, we have introduced three clustering and outlier-based schemes for network anomaly detection and discussed their effectiveness considering several synthetic and real-life datasets. This chapter focuses on DDoS attacks and starts with a description of basics of DDoS attacks, significance of such attacks and detection methods with a general comparison under each category. A distributed denial of service (DDoS) attack [120,375] is a large-scale, coordinated attack on the availability of services of a victim system or network resources, launched indirectly through many compromised computers on the Internet. These attacks normally consume a huge fraction of the resources of a server, making it impossible for legitimate users to access the server. Such attacks also consume excessive network bandwidth by compromising network traffic. These attacks generate a huge surge in traffic with focus on a victim through the intermediary of compromised hosts within a short time interval. A very important requirement of a DDoS attack detection scheme is cost effectiveness and scalability. The scheme should be scalable enough to handle large amount of traffic instantly with high detection accuracy. However, the schemes introduced in the previous chapters are not adequate for this purpose In this chapter, we introduce an effective extended entropy metric-based DDoS flooding attack detection scheme to detect four classes of DDoS attacks, viz , constant rate, pulsing rate, increasing rate and subgroup attacks. The scheme aims

to identify DDoS flooding attacks by measuring the metric difference between legitimate traffic and attack traffic It exploits a generalized entropy metric with packet intensity computation over the sampled network traffic with respect to time We also extend the mechanism into use an ensemble of extended entropy metrics for increasing detection rate in near real-time The proposed scheme is evaluated using several real world DDoS datasets and has been found to outperform the competing methods when detecting classes of DDoS flooding attacks

8.1 Introduction

Due to the increasing use of modern networks and Internet technologies in users' several day-to-day tasks, the network vulnerabilities are also growing exponentially in view of design weaknesses of networks With the recent exponential growth in Internet attacks, it has become crucially important to detect network traffic anomalies including intelligent attacks to keep secure enterprise networks Programs that enable launching of denial of service attacks have been around for many years Old single source attacks are now countered easily by many defense mechanisms and the source(s) of such attacks can be easily rebuffed or shut down with improved tracking capabilities However, with the astounding growth of the Internet during the last decade an increasingly large number of vulnerable systems are now available to attackers Attackers can now employ a large number of these vulnerable hosts to launch an attack instead of using a single server, an approach which was never very effective and detected easily

The first well-documented DDoS attack appears to have occurred in August 1999, when a DDoS tool called Trinoo was deployed in at least 227 systems, to flood a single University of Minnesota computer, which was knocked down for more than two days[1] The first large-scale DDoS attack took place on February 2000[1] On February 7, Yahoo! was the victim of a DDoS attack during which its Internet portal was inaccessible for three hours On February 8, Amazon Buy com, CNN and eBay were all hit by DDoS attacks that caused them to either stop functioning completely or slowed them down significantly[1]

[1] http //www garykessler net/library/ddos html

DDoS attack networks follow two types of architectures the Agent-Handler architecture and the Internet Relay Chat (IRC)-based architecture as discussed by [10 376] The Agent-Handler architecture for DDoS attacks is comprised of clients, handlers, and agents (see Figure 8 1) The attacker communicates with the rest of the DDoS attack system at the client systems The handlers are often software packages located throughout the Internet that are used by the client to communicate with the agents Instances of the agent software are placed in the compromised systems that finally carry out the attack The owners and users of the agent systems are generally unaware of the situation In the IRC-based DDoS attack architecture, an IRC communication channel is used to connect the client(s) to the agents (see Figure 8 2) IRC ports can be used for sending commands to the agents This makes DDoS command packets more untraceable Moreover, it is easier for an attacker to hide his presence in an IRC channel as such channels tend to have large volumes of traffic A recent attacking tool by anonymous based on the IRC protocol is LOIC (Low Orbit Ion Cannon) [377] It includes three primary methods of attacks for TCP UDP, HTTP and is found in two versions *binary* and *web-based* It allows clients to connect remotely via the IRC protocol and to be a part of a system of compromised hosts The bigger the size of compromised hosts the more powerful the attack is In addition to these two architectures, the agent handler architecture is also commonly found in use in the literature [10, 376] Along with the evolution of new DDoS attack tools many DDoS defense mechanisms have also been proposed These approaches are of three types depending on their locality of deployment source-end, victim-end and intermediate network [378] Detecting any DDoS attack at the victim end is easy, but often not useful after legitimate clients have been denied access Source-end detection is a very challenging task Detection approaches used include statistical soft-computing, clustering, knowledge-based and classifiers

DDoS attacks are distributed, cooperative large scale attacks originate and spread in both wired and wireless networks [10, 379] in parallel ways Hence both industry and academia are mostly interested in defending from DDoS attacks and protecting access by legitimate users The detection of DDoS attacks is not an easy task due to the use of forged source addresses and concealment of the attack sources

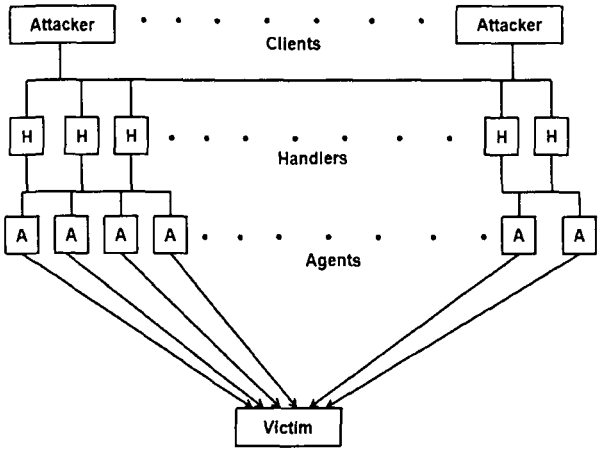


Figure 8.1: Agent-handler network of DDoS attack

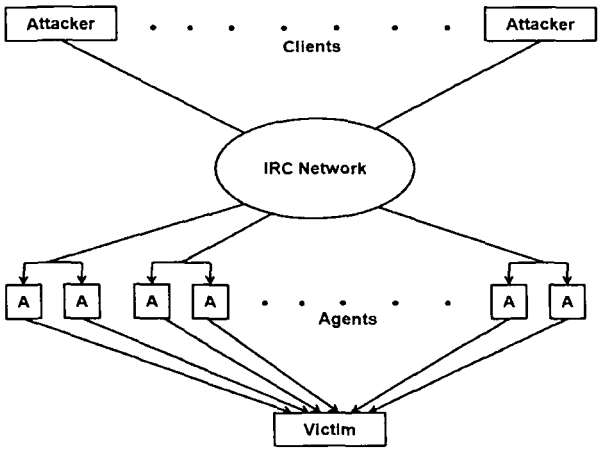


Figure 8.2: IRC-based network of DDoS attack

using several techniques In addition, it is difficult to distinguish attack traffic from normal traffic considering just their traffic rates. An information theory-based network behavior mimicking DDoS attacks detection method is introduced in [380]. It can discriminate mimicked flooding attacks from legitimate access traffic effectively. Several research efforts on DDoS detection [381–383], mitigation [384–386] and filtering [387,388] have been conducted separately. However, the efforts on both detection and IP traceback are limited especially if real-time mitigation is desried. There are two types of DDoS attack based on the flow of traffic rate, viz , (a) high-rate DDoS attack traffic, which is exceptional and (b) low-rate DDoS attack, which is similar to normal traffic [389]. Low-rate DDoS attack is difficult to detect and mitigate within a short time interval

8.1.1 Motivation and Contributions

Network or host-based attack detection methods are of two types: signature-based and anomaly-based. A signature-based method builds profiles using known characteristics of both attack and normal traffics, and then matches the incoming traffic with it to report any alarm. In contrast, the anomaly based method models the normal behavior and compares it with incoming traffic for any deviation. Several information theory based metrics have been proposed to overcome the problems faced by both misuse and anomaly detection methods [379, 390]. Information theory can associate an uncertainty measure with a random variable. Entropy is the commonly used property because its value depends on the amount of material or information. Shannon and Renyi entropies [391] share the property that the joint entropy of a pair of independent random variables equals the sum of individual entropies Shannon's entropy and Kullabck Leibler Divergence have both been regarded as effective methods to detect abnormal traffic based on IP address distribution statistics or packet size distribution statistics [392]. For any DDoS defense system, the main criteria to achieve are (a) early stage detection, (b) high accuracy and (c) low false alarm rate. Researchers have failed to achieve all these goals simultaneously. The following major contributions have been made in this chapter

- We present a survey of DDoS attacks, a taxonomy, detection methods and tools. In our taxonomy, there are seven distinct possibilities in which an intruder can attempt to launch DDoS attacks. We include a detailed discussion of various DDoS defense mechanisms and methods under the broad categories of statistical, knowledge-based, soft computing, data mining, and machine learning.

- We propose an effective DDoS flooding attack detection scheme using an extended entropy metric that adapts the generalized entropy with packet intensity in a sampled traffic within a time interval. We attempt to detect four classes of DDoS flooding attacks [392], viz., constant rate, pulsing rate, increasing rate and subgroup attacks obtained based on attack rate dynamics (see Figure 8.3).

- We also extend our scheme to use an ensemble to increase detection rate in

near real-time. The proposed scheme demonstrates the effective increase of detection rate while detecting major classes of DDoS flooding attacks.

- We present extensive experimental results using real-world DDoS datasets. These include (i) the MIT Lincoln Laboratory dataset, (ii) the CAIDA DDoS 2007 dataset and (iii) the TUIDS DDoS dataset as discussed in Chapter 4. The performance of the proposed scheme is superior in comparison to competing methods.

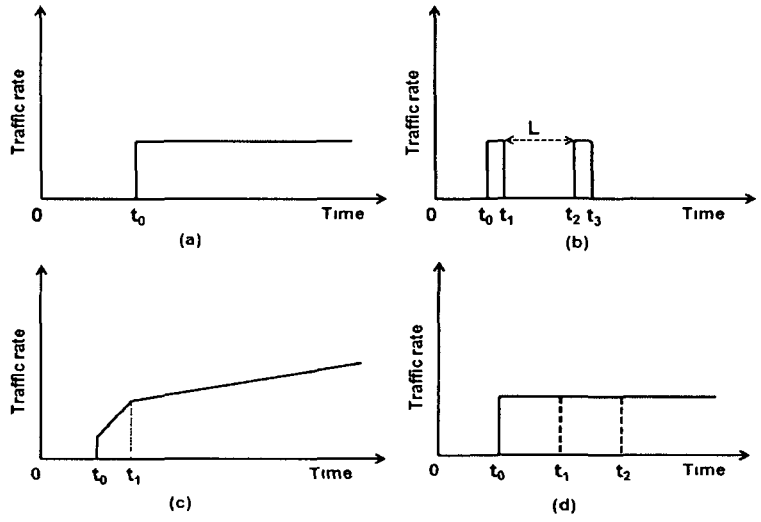


Figure 8.3: Classes of DDoS flooding attacks based on the attack rate dynamics: (a) constant rate, (b) pulsing rate, (c) increasing rate and (d) subgroup attack

8.2 DDoS Attack and Related Concepts

As stated in [10,393], a DDoS attack can be defined as an attack which uses a large number of compromised computers to launch a coordinated DoS attack against a single machine or multiple victim machines. Using client/server technology, the perpetrator is able to multiply the effectiveness of the DoS attack significantly by harnessing the resources of multiple unwitting accomplice computers, which serve as attack platforms. Approximate attack statistics for DDoS [394] up to the year 2013 are shown in Figure 8.4. A DDoS attacker is considered more intelligent than a DoS attacker. It is distinguished from other attacks by its ability to deploy its weapons in a "distributed" way over the Internet and to aggregate these forces to

create lethal traffic. Rather than breaking the victim's defense system for fun or to show prowess, a DDoS attack usually aims to cause damage on a victim either for personal reasons and material gain although some attacks may be to gain popularity.

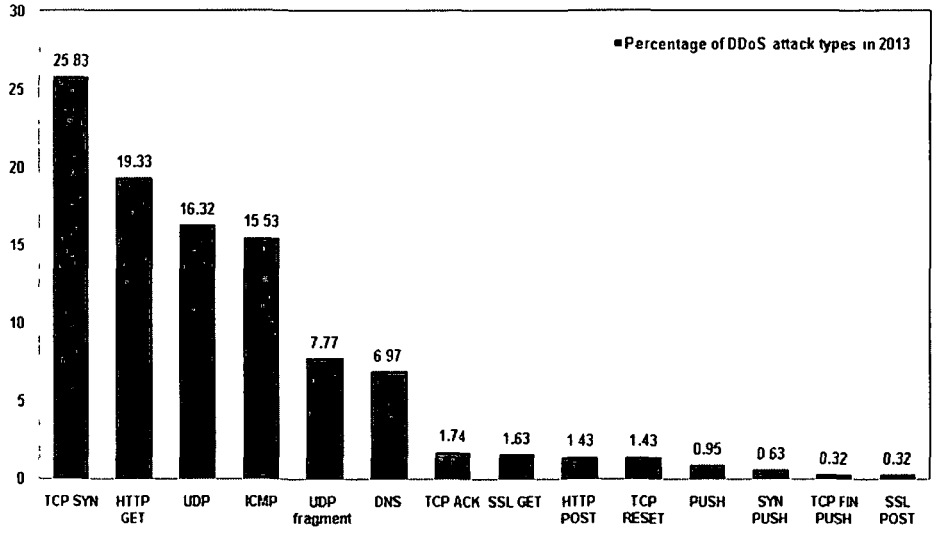


Figure 8.4: DDoS attacks statistics up to the year 2013 [source: [9]]

DDoS attacks mainly take advantage of the architecture of the Internet and this is what makes them powerful. While designing the Internet, the prime concern was to provide for functionality, not security. As a result, many security issues have been raised, which are exploited by attackers. Some of the issues are given below.

- Internet security is highly interdependent. No matter how secure a victim's system may be, whether or not this system will be a DDoS victim depends on the rest of the global Internet [395, 396].
- Internet resources are limited. Every Internet host has limited resources that sooner or later can be exhausted by a sufficiently large number of users.
- Many against a few: If the resources of the attackers are greater than the resources of the victims, the success of the attack is almost definite.
- Intelligence and resources are not collocated. Most intelligence needed for service guarantees is located at end hosts. At the same time high bandwidth pathways needed for large throughput are situated in the intermediate network. Such abundant resources present in unwitting parts of the network are exploited by the attacker to launch a successful flooding attack.

- The handlers or the masters, which are compromised hosts with special programs running on them, are capable of controlling multiple agents
- The attack daemon agents or zombie hosts are compromised hosts that are running a special program each and are responsible for generating a stream of packets towards the intended victim These machines are commonly external to the victim's own network to disable efficient response from the victim, and external to the network of the attacker to forswear liability if the attack is traced back

8.2.1 DDoS Strategy

A Distributed Denial of Service (DDoS) attack is composed of several elements as shown in Figures 8 5 and 8 6

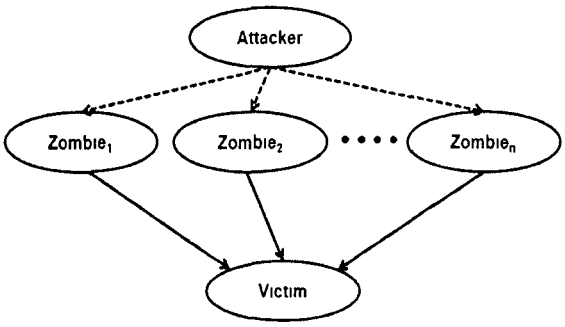


Figure 8.5: *Direct DDoS attack* Send control traffic directly to the zombies to attack the victim host

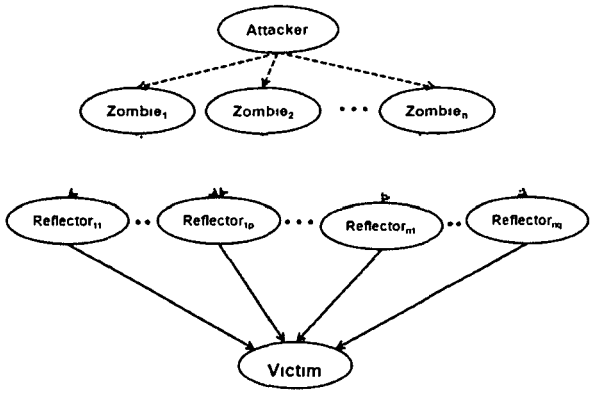


Figure 8.6: *Indirect DDoS attack* Send control traffic indirectly to the zombies to compromise the target host Reflectors are non-compromised systems that exclusively send replies to a request

There are four basic steps in launching a DDoS attack. These are shown in Figure 8.7.

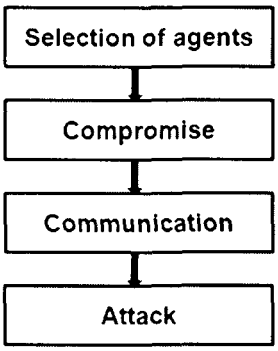


Figure 8.7: Steps to perform a DDoS attack

(a) *Selection of agents* The attacker chooses the agents that will perform the attack. Based on the nature of vulnerabilities present, some machines are compromised to use as agents. Attackers victimize these machines, which have abundant resources, so that a powerful attack stream can be generated. In early years, the attackers attempted to acquire control of these machines manually. However, with the development of advanced security attack tool(s), it has become easier to identify these machines automatically and instantly.

(b) *Compromise.* The attacker exploits security holes and vulnerabilities of the agent machines and plants the attack code. Not only that, the attacker also takes necessary steps to protect the planted code from identification and deactivation. As per the direct DDoS attack strategy, shown in Figure 8.5, the compromised nodes, i.e , zombies between the attacker and victim are recruited unwitting accomplice hosts from a large number of unprotected hosts connected through the Internet in high bandwidth On the other hand, the DDoS attack strategy shown in Figure 8.6 is more complex due to inclusion of intermediate layer(s) between the zombies and victim(s). It further complicates the traceback mostly due to (i) complexity in untangling the traceback information (partial) with reference to multiple sources, and/or (ii) having to connect a large number of routers or servers. Self-propagating tools such as the Ramen worm [397] and Code Red [398] automate this phase Unless a sophisticated defense mechanism is used, it is usually difficult for the users and owners of the agent systems to

realize that they have become a part of a DDoS attack system. Another important feature of such an agent system is that the agent programs are very cost effective both in terms of memory and bandwidth. Hence they affect the performance of the system minimally.

(c) *Communication* The attacker communicates with any number of handlers to identify which agents are up and running, when to schedule attacks or when to upgrade agents. Such communications among the attackers and handlers can be via various protocols such as ICMP, TCP or UDP. Based on configuration of the attack network, agents can communicate with a single handler or multiple handlers.

(d) *Attack* The attacker initiates the attack. The victim, the duration of the attack as well as special features of the attack such as the type, length TTL, and port numbers can be adjusted. The attackers use available bandwidth and send huge number of packets to the target host or network to overwhelm the resources immediately.

8.2.2 DDoS Attack Taxonomy

A taxonomy of DDoS attacks based on [392] is given in Figure 8.8. We see in the taxonomy that intruders attempt to launch DDoS attacks based on exploitation of various means (shown in the left column) and their resultant effects can be observed at various levels.

8.2.3 Architecture of DDoS Attack Defense Mechanisms

Based on the locality of deployment, DDoS defense schemes can be divided into three classes [378]: victim-end, source-end and intermediate network defense mechanisms. All of these mechanisms have their own advantages and disadvantages. We discuss them one by one.

By degree of automation	Manual	
	Semiautomatic	Direct
		Indirect
	Automatic	
By exploited vulnerability	Flood attack	Application level flood
		Network level flood
	Amplification attack	Smurf attack
		Fraggle attack
	Protocol exploit attacks	
	Malformed packet attack	
By attack network used	Attacks based on an agent handler network	
	IRC Botnet based	
	Peer to peer network based	
By attack rate dynamics	Continuous	
	Variable	Fluctuating
		Increasing
By victim type	Host	
	Resource	
	Network	
	Application	
By impact	Disruptive	Recoverable
		Phlashing
	Degrading	
By agent set	Constant agent set attacks	
	Variable set attacks	

Figure 8.8: A taxonomy of DDoS attacks [10]

Victim-end Defense Mechanism

In the victim-end defense mechanism detection and response are generally done in the routers of victim networks, i e , networks providing critical Internet services These mechanisms can closely observe the victim network traffic, model its behavior and detect anomalies Detecting DDoS attacks in victim routers is relatively easy because of the high rate of resource consumption It is also the most practically applicable type of defense mechanism that can classify the attack traffic from legitimate traffic But the main problems with this mechanism are (i) During DDoS attacks, victim resources, e g network bandwidth, often get overwhelmed and cannot stop the flow beyond victim routers and (ii) It can detect the attack only after it reaches the victim and detecting an attack when legitimate clients have already been denied is not useful A generic architecture of such mechanisms is shown in Figure 8 9

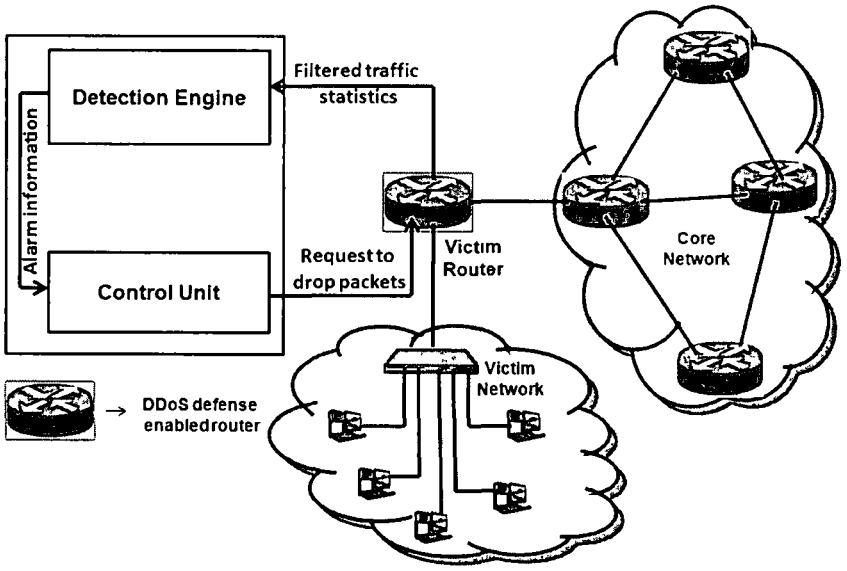


Figure 8.9: Generic architecture for victim-end DDoS defense mechanism

Source-end Defense Mechanism

Detecting and stopping a DDoS attack at the source is made in the source-end defense mechanism. This mechanism detects malicious packets and prevents the possibility of flooding but not at the victim side. It is best to filter or rate limit malicious traffic with minimum damage within the legitimate traffic Moreover, a source-end based defense mechanism gains knowledge from a small amount of traffic and consumes minimum resources (i.e., processing power and buffer). The main difficulties of this mechanism are: (i) It cannot observe suspicious traffic at the victim-end because it has no interaction with the victim node, (ii) Sources are widely distributed and a single source behaves almost similarly as in normal traffic, and (iii) Identification of deployment points are at the source-end. A generic architecture of source-end defense mechanisms is shown in Figure 8.10.

Intermediate Network Defense Mechanism

The intermediate network defense scheme balances the trade-offs between detection accuracy and attack bandwidth consumption, the main issues in source-end and victim-end detection mechanisms. It can be deployed in any network router connected to an ISP. Such a scheme is generally collaborative in nature and the routers share their observations with other routers Detection of attack sources is

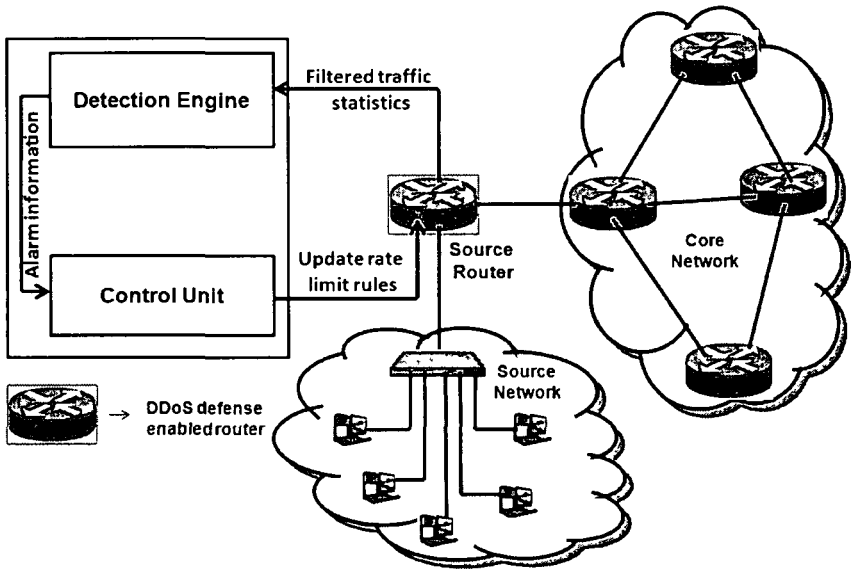


Figure 8.10: Generic architecture for source-end-based DDoS defense mechanism

easy in this approach due to collaborative operation. Routers can form an overlay mesh to share their observations [399]. The main difficulty with this mechanism is the location of deployment. The unavailability of this mechanism in only a few routers may cause failure to the detection effort and the full practical implementation of this mechanism is extremely difficult because it will require reconfiguring all the routers on the Internet Figure 8 11 shows a generic architecture of the intermediate network defense mechanism

To address the above deficiencies it would be beneficial to construct victim-end based defense mechanisms that can perform detection and IP traceback of DDoS attacks with a low false positive rate, performed within a short time interval. A comparison of DDoS defense mechanisms at different deployment locations is given in Table 8.1. From the table, it can be observed that victim-end system is advantageous in view of the following points.

Table 8.1: Feasibility of DDoS defense at deployment location

Deployment	Characteristics	Rate limiting/ Filtering	Defense vulnerability/ Robustness	Deployment difficulty
Source-end	Very difficult	Easy	Low	Highly difficult
Victim-end	Easy	Difficult	High	Very easy
Intermediate network	Difficult	Difficult	Medium	Difficult

- It can closely observe the victim system or host to analyze the network traffic

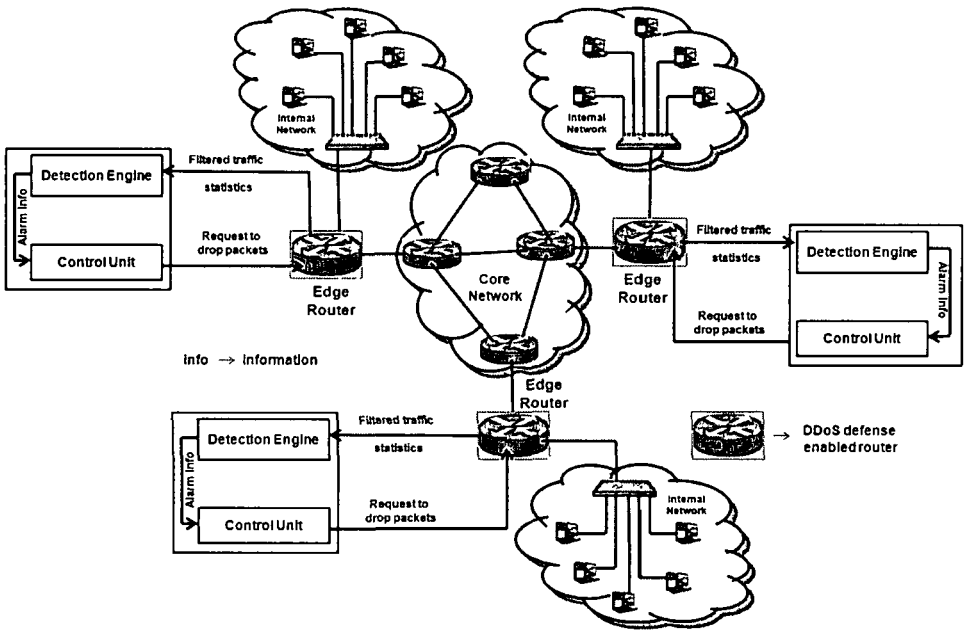


Figure 8.11: Generic architecture for intermediate network-based DDoS defense mechanism

in near real-time.

- Easy to deployment.

- Cheaper to detect DDoS attacks than other mechanisms.

8.3 Prior Research

In this section, we present a summary of existing literature on DDoS attack detection methods. These methods are based on the three architectures introduced earlier namely, victim-end, source-end and intermediate network. We discuss these schemes without presenting how they can be deployed in real networks. Recent trends show that soft computing approaches we being widely used heavily for DDoS attack detection. Ensembles of classifiers have also performed satisfactorily with high detection rates. We classify methods for DDoS attack detection into four major classes as shown in Figure 8.12

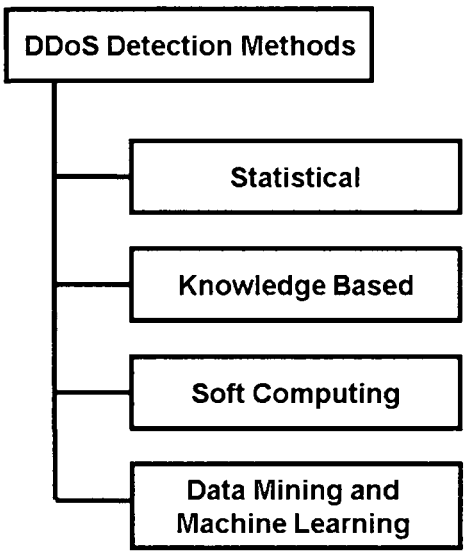


Figure 8.12: Classification of DDoS attack detection methods

8.3.1 Statistical Methods

Statistical properties of normal and attack patterns can be exploited to detect DDoS attacks. Generally a statistical model for normal traffic is fitted and then a statistical inference test is applied to determine if a new instance belongs to this model. Instances that do not conform to the learnt model, based on the applied test statistics, are classified as anomalies. Chen et al. [400] develop a distributed change point (DCP) detection architecture using change aggregation trees (CATs). The non-parametric CUSUM approach was adapted to describe the distribution of pre-change or post-change network traffic When a DDoS flooding attack is being launched, the cumulative deviation is noticeably higher than random fluctuations. The CAT mechanism is designed to work at the router level to detect abrupt changes in traffic flows. The domain server uses the traffic change patterns detected at attack-transit routers to construct the CATs, which represent the attack flow patterns A very well-known DDoS defense scheme called D-WARD is presented in [401]. D-WARD identifies an attack based on continuous monitoring of bidirectional traffic flows between the network and the rest of the Internet and by periodic deviation analysis with the normal flow patterns. D-WARD not only offers a good detection rate but also reduces DDoS attack traffic significantly.

Saifullah [402] proposes a defense mechanism based on a distributed algorithm

that performs weight-fair throttling at upstream routers. The throttling is weight-fair because the traffic destined for the server is controlled (increased or decreased) by leaky buckets at the routers based on the number of users connected, directly or through other routers, to each router In the beginning of the algorithm, the survival capacity is underestimated by the routers so as to protect the server from any sudden initial attack The rate is updated (increased or decreased), based on the server's feedback sent to its child routers and eventually propagated downward to all routers, in the subsequent rounds of the algorithm with a view to converging the total server load to the tolerable capacity range. Feinstein and Dan [403] present methods to identify DDoS attacks using entropy computation and frequency-sorted distribution of relevant packet attributes They show DDoS attacks as anomalies and demonstrates their performance on real network traffic traces obtained from a variety of network scenarios.

Akella et al. [404] explore key challenges in helping an ISP network detect attacks on itself or attacks on external sites which use the ISP network. They propose a detection mechanism where each router detects traffic anomalies using profiles of normal traffic constructed using stream sampling algorithms. Initial results show that it is possible to (a) profile normal traffic reasonably accurately, (b) identify anomalies with low false positive and false negative rates (locally, at the router) and (c) still be cost effective in terms of memory consumption and per packet computation. In addition, ISP routers exchange information with one another to increase confidence in their detection decisions A router gathers responses from all other routers regarding suspicions and based on these responses decides whether a traffic aggregate is an attack or is normal. The initial results show that individual router profiles capture key characteristics of the traffic effectively and identify anomalies with low false positive and false negative rates Peng et al. [405] describe a novel approach to detect bandwidth attacks by monitoring the arrival rate of new source IP addresses. The detection scheme is based on an advanced non-parametric change detection scheme, CUSUM. Cheng et al. [406] propose the IP Flow Feature Value (FFV) algorithm based on the essential features of DDoS attacks such as abrupt traffic change, flow dissymmetry, distributed source IP addresses and concentrated target IP addresses. Using a linear prediction technique,

a simple and efficient ARMA prediction model is established for both normal and attack network flows Udhayan and Hamsapriya [407] present a Statistical Segregation Method (SSM), which samples the flow in consecutive intervals and compares the samples against the attack state condition and sorts them with the mean as the parameter. Then correlation analysis is performed to segregate attack flows from legitimate flows The authors compare SSM against various other methods and identify a blend of segregation methods for alleviating false detections A brief summary of these methods is given in Table 8.2

Table 8.2: Statistical DDoS attack detection methods

Reference	Objective	Deployment	Mode of working	Trainable	Remarks
Mirkoviac et al [401]	Attack prevention	Source side	Centralized	Yes	Detects DDoS attacks at the source end autonomously and stops attacks from the source network using statistical traffic modeling
Akella et al [404]	Attack detection	Source and victim side	Distributed	No	Detects traffic anomalies in router using stream sampling algorithms based on profiles constructed from normal traffic
Peng et al [405]	Detecting bandwidth attacks	Victim side	Centralized	Yes	Uses sequential nonparametric change point detection method to improve the accuracy
Chen et al [400]	Attack detection and traceback	Between source and destination network	Distributed	No	Automatically performs traceback during the detection of suspicious traffic flows
Saifullah [402]	Attack prevention	Between source and destination network	Distributed	No	Protects Internet server from DDoS attacks using distributed weight-fair throttling at the upstream routers
Cheng et al [406]	Attack detection	Victim side	Centralized	Yes	Exploits four flow features burst in the traffic volume, asymmetry of the flow distributed source IP addresses and concentrated destination IP address while detecting DDoS attacks
Udhayan and Hamsapriya [407]	minimize false alarm	Victim side	Centralized	Yes	Uses a statistical segregation method for detecting DDoS attacks based on sampling of flow in consecutive time interval

8.3.2 Soft Computing Methods

Learning paradigms, such as neural networks. radial basis functions and genetic algorithms are increasingly used in DDoS attack detection because of their ability to classify intelligently and automatically. Soft computing is a general term for describing a set of optimization and processing techniques that are tolerant of imprecision and uncertainty Jalili et al. [408] introduce a DDoS attack detection system called SPUNNID based on a statistical pre-processor and unsupervised ar-

tificial neural nets They use statistical pre-processing to extract features from the traffic and an unsupervised neural net to analyze and classify traffic patterns as either a DDoS attack or normal

Karimazad and Faraahi [409] propose an anomaly-based DDoS detection method based on features of attack packets, analyzing them using Radial Basis Function (RBF) neural networks The method can be applied to edge routers of victim networks Vectors with seven features are used to activate an RBF neural network at each time window The RBF neural network is applied to classify data to normal and attack categories If the incoming traffic is recognized as attack traffic, the source IP addresses of the attack packets are sent to the Filtering Module and the Attack Alarm Module for further actions Otherwise, if the traffic is normal, it is sent to the destination RBF neural network training can be performed as an off-line process but it is used in real time to detect attacks faster Nguyen and Choi [410] develop a method for proactive detection of DDoS attacks by classifying the network status They break a DDoS attack into phases and select features based on an investigation of real DDoS attacks Finally, they apply the k-nearest neighbor (KNN) method to classify the network status in each phase of the DDoS attack A method presented in [383] detects DDoS attacks based on a fuzzy estimator using mean packet inter-arrival times It detects the suspected host and traces the IP address to drop packets within 3 second detection windows

Lately, ensembles of classifiers have been used for DDoS attack detection The use of an ensemble reduces the bias of existing individual classifiers An ensemble of classifiers has been used by [411] for this purpose where a Resilient Back Propagation (RBP) neural network is chosen as the base classifier The main focus of this work is to improve the performance of the base classifier The proposed classification algorithm, RBPBoost combines the output of the ensemble of classifiers and the Neyman Pearson cost minimization strategy [412] for final classification decision Table 8 3 presents a brief summary of the soft computing methods presented in this section

Table 8.3: Soft computing-based DDoS attack detection methods

Reference	Objective	Deployment	Mode of working	Trainable	Remarks
Jalili et al [408]	Attack detection	Victim side	Centralized	Yes	Uses statistical preprocessor and unsupervised neural network classifier for DDoS attack detection
Nguyen and Choi [410]	Attack detection	Intermediate network	Centralized	Yes	Detects only known attacks using k-nearest neighbor based technique
Karimazad and Faraahi [409]	Attack detection	Victim side	Centralized	Yes	Uses Radial Basis Function (RBF) neural networks and gets low false alarm rate
Kumar and Selvakumar [411]	Attack detection	Victim side	Centralized	Yes	RBPBoost combines an ensemble of classifier outputs and Neyman Pearson cost minimization strategy for final classification decision during DDoS attack detection and gets high detection rate

8.3.3 Knowledge-based Methods

In knowledge-based approaches, network events are checked against predefined rules or patterns of attack. In these approaches, general representations of known attacks are formulated to identify actual occurrences of attacks. Examples of knowledge-based approaches include expert systems, signature analysis, self organizing maps, and state transition analysis Gil and Poletto [413] introduce a heuristic along with a data structure called MULTOPS (MUlti-Level Tree for Online Packet Statistics) that monitors certain traffic characteristics which can be used by network devices such as routers to detect and eliminate DDoS attacks MULTOPS is a tree of nodes that contains packet rate statistics for subnet prefixes at different aggregation levels. Expansion and contraction of the tree occurs within a pre-specified memory size. A network device using MULTOPS detects ongoing bandwidth attacks by the presence of a significant and disproportional difference between packet rates going to and coming from the victim or the attacker. Depending on their setup and their location on the network, MULTOPS-equipped routers or network monitors may fail to detect a bandwidth attack that randomizes IP source addresses on malicious packets. MULTOPS fails to detect attacks that deploy a large number of proportional flows to cripple a victim.

Thomas et al. [414] present an approach to DDoS defense called NetBouncer and claim it to be a practical approach with high performance. Their approach relies on distinguishing legitimate and illegitimate uses and ensuring that resources are made available only for legitimate use NetBouncer allows traffic to flow with

reference to a long list of proven legitimate clients. If packets are received from a client (source) not on the legitimate list, a NetBouncer device proceeds to administer a variety of legitimacy tests to challenge the client to prove its legitimacy. If a client can pass these tests, it is added to the legitimate list and subsequent packets from the client are accepted until a certain legitimacy window expires.

Wang et al. [415] present a formal and methodical way to model DDoS attacks using Augmented Attack Trees (AAT) and discuss an AAT-based attack detection algorithm This model explicitly captures the subtle incidents triggered by a DDoS attack and the corresponding state transitions considering network traffic transmission on the primary victim server. Two major contributions of this work are (1) an AAT-based DDoS model (ADDoSAT), developed to assess potential threats from malicious packets on the primary victim server and to facilitate the detection of such attacks and (2) an AAT-based bottom-up detection algorithm proposed to detect all kinds of attacks based on AAT modelling. Compared to the conventional attack tree modelling method, AAT is more advanced because it provides additional information, especially about the state transition process. As a result, it overcomes the shortcomings of CAT modelling There is currently no established AAT-based bottom-up procedure for detecting network intrusions. Limwiwatkul and Rungsawang [416] discover DDoS attack signatures by analyzing TCP/IP packet headers against well-defined rules and conditions and distinguishing the difference between normal and abnormal traffic. The authors mainly focus on ICMP, TCP and UDP flooding attacks.

Zhang and Parashar [32] propose a distributed approach to defend against DDoS attacks by coordinating across the Internet. Unlike traditional IDS, it detects and stops DDoS attacks within the intermediate network. In the proposed approach, DDoS defence systems are deployed in the network to detect DDoS attacks independently. A gossip based communication mechanism is used to exchange information about network attacks between these independent detection nodes to aggregate information about the overall network attacks. Using the aggregated information, individual defence nodes obtain approximate information about global network attacks and can stop them more effectively and accurately. For faster and reliable dissemination of attack information, the network grows as a peer-to-peer overlay

network on top of the Internet. Previously proposed approaches rely on monitoring the volume of traffic that is received by the victim. Most such approaches are incapable of differentiating a DDoS attack from a flash crowd. Lu et al [382] describe a perimeter-based anti-DDoS system, in which the traffic is analyzed only at the edge routers of an Internet Service Provider (ISP) network. The anti-DDoS system consists of two major components: (1) temporal-correlation based feature extraction and (2) spatial-correlation based detection. The scheme can accurately detect DDoS attacks and identify attack packets without modifying existing IP forwarding mechanisms at the routers. A brief summary of these knowledge based methods is given in Table 8 4

Table 8.4: Knowledge based DDoS attack detection methods

Reference	Objective	Deployment	Mode of working	Trainable	Remarks
Gil and Poletto [41 ‡]	Attack prevention	Between source and destination network	Centralized	No	Each network devices maintains a data structure known as MULTOPS Fails to detect attacks that deploy a large number of DDoS attack flows using a large number of agents, IP spoofing attacks
Thomas et al [414]	Attack detection	Victim side	Centralized	No	NetBouncer differentiates DDoS traffic from flash crowd using inline packet processing based on network processor technology
Limwiwatkul and Rungsawang [416]	Attack detection	Victim side	Distributed	Yes	Uses a TCP packet header to construct attack signature model for DDoS attack detection
Zhang and Parashar [32]	Proactive	Intermediate network	Distributed	Yes	A gossip based scheme uses to get global information about DDoS attacks by information sharing
Lu et al [382]	Attack detection	Edge router	Distributed	Yes	Exploits spatial and temporal correlation of DDoS attack traffic records for detecting anomalous packets
Wang et al [415]	Attack detection	Victim side	Centralized	No	Uses an Augmented Attack Tree model for the detection of DDoS attacks and also can detect other attacks

8.3.4 Data Mining and Machine Learning Methods

An effective defense system to protect network servers, network routers and client hosts from becoming handlers, zombies and victims of DDoS flooding attacks is presented in [417] The NetShield system can protect any IP-based public network on the Internet. It uses preventive and deterrent controls to remove system vulnerabilities on target machines. Adaptation techniques are used to launch protocol anomaly detection and provide corrective intrusion responses. The NetShield system enforces dynamic security policies. NetShield is especially tailored to protect

network resources against DDoS flooding attacks Lee et al [418] propose a method for proactive detection of DDoS attacks by exploiting an architecture consisting of a selection of handlers and agents that communicate, compromise and attack The method performs cluster analysis. The authors experiment with the DARPA 2000 Intrusion Detection Scenario Specific Dataset to evaluate the method. The results show that each phase of the attack scenario is partitioned well and can detect precursors of a DDoS attack as well as the attack itself. Sekar et al [419] investigate the design space for in-network DDoS attack detection and propose a triggered, multi-stage approach that addresses both scalability and accuracy. Their contribution is the design and implementation of LADS (Large-scale Automated DDoS detection System) The system makes effective use of the data (such as NetFlow and SNMP feeds from routers) readily available to an ISP.

A two-stage automated system is proposed in [420] to detect DoS attacks in network traffic. It combines the traditional change point detection approach with a novel one based on continuous wavelet transforms [421]. The authors test the system using a set of publicly available attack-free traffic traces superimposed with anomaly profiles. Li and Lee [422] present a systematic wavelet based method for DDoS attack detection. They use energy distribution based on wavelet analysis to detect DDoS attack traffic. Energy distribution over time has limited variation if the traffic keeps its behavior over time. The method presented in [423] can identify flooding attacks in real time and also can assess the intensity of the attackers based on fuzzy reasoning. The process consists of two stages (i) statistical analysis of the network traffic time series using discrete wavelet transform and Schwarz information criterion (SIC) to find the change point of the Hurst parameters resulting from a DDoS flood attack, and then (ii) identification and assessment of the intensity of the DDoS attack adaptively based on an intelligent fuzzy reasoning mechanism. Test results by ns2[1] based simulation with various network traffic characteristics and attack intensities demonstrate that the method can detect DDoS flood attacks on time, effectively and intelligently Zhang et al. [388] present a CPR (Congestion Participation Rate) based approach to detect low-rate DDoS (LDDoS) attacks using flow level network traffic. A flow with a higher CPR value leads to LDDoS and

[1] http://www isi.edu/nsnam/ns/

consequent dropping of the packets. The authors evaluate the mechanism using ns2 simulation, testbed experiments and Internet traffic trace and claim that the method can detect LDDoS flows effectively

In [424], a mathematical model is presented to provide gross evaluation of the benefits of DDoS defence based on dropping of attack traffic. Simulation results and testbed experiments are used to validate the model. In the same work, the authors also consider an autonomic defence mechanism based on CPN (Cognitive Packet Network) protocol and establish it to be capable of tracing back flows coming into a node automatically Yuan and Kevin [425] present a DDoS flooding attack detection scheme by monitoring network-wide macroscopic effects. They work with several attack modes including constant rate, increasing rate, pulsing rate and subgroup attacks. Lee and Xiang [177] describe several information theoretic measures for anomaly detection. These are entropy, conditional entropy, information gain and information cost, tested on several datasets. A summarized presentation of these methods in this category is given in Table 8.5.

A low-rate DDoS attack has significant ability to conceal its traffic because of its similarity with normal traffic. Xiang et al. [379] propose two new information metrics: (i) generalized entropy metric and (ii) information distance metric, to detect low-rate DDoS attacks. They identify the attack by measuring the distance between legitimate traffic and attack traffic. The generalized entropy metric is more effective than the traditional Shannon metric [426]. In addition, the information distance metric outperforms the popular Kullback-Leibler divergence approach. Francois et al. [427] present a method called *FireCol* based on information theory for early detection of flooding DDoS attacks. FireCol is comprised of an intrusion prevention system (IPS) located at the Internet Service Provider (ISP) level. The IPSs form virtual protection rings around the hosts to defend and collaborate by exchanging selected traffic information. An entropy variation based detection and IP traceback scheme for DDoS attacks is proposed at [399]. The system observes and stores short-term information on flow entropy variations at routers. Once the detection algorithm detects a DDoS attack, it initiates the pushback tracing procedure to find the actual location of attacks. Wei et al. [428] propose a Rank Correlation-based Detection (RCD) algorithm for detecting distributed reflection

DoS attacks. Preliminary simulations show that RCD can differentiate reflection flows from legitimate ones effectively.

Table 8.5: Data mining and machine learning-based DDoS attack detection methods

Reference	Objective	Deployment	Mode of working	Trainable	Remarks
Hwang et al [417]	Attack prevention	Victim-end	Centralized	Yes	Protects network servers, routers and clients from DDoS attacks using protocol anomaly detection technique
Li and Lee [422]	Attack detection	Victim-end	Centralized	No	An energy distribution based wavelet analysis technique for the detection of DDoS traffic
Sekar et al [419]	Attack detection	Source-end	Distributed	Yes	A triggered multi-stage approach for both scalability and accuracy for DDoS attack detection
Gelenbe and Loukas [424]	Attack defense using packet dropping	Victim-end	Centralized	Yes	Detects attack by tracing back flows automatically
Lee et al [418]	Attack detection	Source-end	Centralized	Yes	Detects DDoS attack proactively based on cluster analysis with agent handler architecture
Dainotti et al [420]	Detection of DoS attack	Victim-end	Centralized	Yes	Detects attacks correctly using combination of traditional change point detection and continuous wavelet transformation
Xia et al [423]	Detects flood attack and its intensity	Victim-end	Centralized	Yes	A method to detect DDoS flooding attack using fuzzy logic
Xiang et al [379]	Detects low rate flooding attacks	Victim-end	Centralized	No	Detects low-rate DDoS flooding attacks using new information metrics effectively
Francois et al [427]	DDoS flooding attack detection	Source-end	Distributed	No	A complete DDoS flooding attack detection technique Also support incremental deployment in real network

8.3.5 Discussion

Exact comparison of DDoS attack detection schemes is not feasible because some works do not specify their results clearly whereas others evaluate their schemes using different datasets or in different testing conditions. A comparison (as shown in Table 8.6) establishes that most works do not consider all the issues that are pertinent. For example, the Distributed Change Point detection method [400] performs well for TCP SYN attacks, but its performance degrades for UDP attacks with large packet sizes D-WARD [401] fails to detect pulsing attacks, especially when the inactive period is large In case of NetBouncer [414], the legitimacy tests may not be exhaustive and certain illegitimate clients may also pass the test. In addition, Netbouncer is overwhelmed by flash crowds. Moreover, the delay introduced by the test affects new legitimate clients. Detection using RBF Neural Networks and sta-

tistical features [409] performs well for known attacks, but no dynamic modification can be performed easily for unknown attacks. The following are some observations.

- It is important to understand the features of DDoS attacks, but it is critical to find effective features to detect an attack.
- Most existing schemes are focused on detecting DDoS attacks with high detection accuracy or low false alarms, but often these methods have been found fail to perform in real-time or near real-time.
- Some schemes are composed of several small modules [403] Due to the lack of timely coordination among them, the total cot increases considerably.
- Even though several information theoretic measures are available [177], it is a difficult task to build an adaptive model to detect DDoS attacks by dynamically adjusting different parameters.
- The basic entropy-based measure produces a high false alarm rate due to low spacing between attack traffic sample and normal traffic sample.
- Only a few ensemble-based methods to detect DDoS attacks have been reported in the literature [429] The reported methods suffer from high false alarm rate

8.4 Problem Statement

We define the problem of DDoS attack detection as follows The problem is to detect DDoS flooding attacks using a minimum and relevant subset of packet features by computing difference of information theoretic distance between real-life attack traffic and legitimate traffic within and relative sample periods. We assume a sample S_i to be anomalous if (a) $S_i \in S$ and $|E(S_i) - E(S_j)| \geq \omega_1$, where E is the information distance metric, S_i and S_j are samples within a sampling period S, ω_1 is the user defined threshold for maximum allowable local entropy variation and (b) $|E(S_p) - E(S_q)| \leq \omega_2$, where S_p and S_q are the relative samples, ω_2 is a user defined threshold for minimum allowable global entropy variation.

Table 8.6: A general comparison of DDoS attack detection methods

Scheme	Approach	Architecture/ Method	w	x	y	z	DR with dataset used	FAR with dataset used
DCD approach [400]	statistical	DCP / Change aggregation tree	R	Yes	Yes	No	98% (flooding attacks)	< 1% (flooding attacks)
Weighted fair model [402]	statistical	Weight Fair Throttling	R	Yes	Yes	No	-	-
SPUNNID model [408]	statistical	SPUNNID system / unsupervised neural network	R	Yes	Yes	Yes	94 9% (flooding attacks)	5% (flooding attacks)
D-WARD system [401]	knowledge based	Self regulating reverse feedback system / rate limited	R	No	Yes	No	(flooding attacks)	0 5% (flooding attacks)
MULTOPS system [413]	knowledge based	Multi-level tree based method	N	Yes	Yes	No	(IP spoofing attacks)	-
NetBouncer model [414]	knowledge based	Packet filtering method	R	Yes	Yes	No	(real-time data)	-
RBF neural net model [409]	soft computing	RBF system / Radial basis function	R	No	No	No	98 2% (UCLA data)	0 01% (UCLA data)
Attack tree model [415]	knowledge based	AATBD system / Tree based	R	No	No	No	(flooding attacks)	-
Signature discovery approach [416]	knowledge based	Traffic statistics based method	R	No	No	No	(flooding attacks)	-
Profile based approach [404]	statistical	Stream sampling based method	R	No	No	No	(IP spoofing attacks)	2% (IP spoofing attacks)
Cooperative model [32]	knowledge based	RL-DDoS system / Gossip-based scheme	R	No	No	No	(Emulab simulation)	7%-12% (Emulab simulation)
Sequential non-parametric change point method [405]	knowledge based	Nonparametric CPD method	R	yes	yes	No	90%-100% (Aucland traces)	-
Perimeter based anti-DDoS system [382]	knowledge based	Spatial correlation based method	R	Yes	No	No	93 0% (IP spoofing attacks)	0 05% (IP spoofing attacks)
K-NN classifier approach [410]	statistical	Nearest neighbor-based method	R	No	No	No	91 88% (DARPA 2000 DDoS data)	8 11% (DARPA 2000 DDoS data)
Change point detect by fuzzy logic [423]	Soft computing	Fuzzy logic based method	R	No	No	No	(ns2 simulation)	-
Linear prediction model [406]	statistical	Linear prediction based method	R	Yes	No	No	96 1% (DDoS flow data, LLDoS 2 0 2)	0 8% (DDoS flow data, LLDoS 2 0 2)
SSM method [407]	statistical	Statistical segregation based method	R	No	No	No	(CAIDA data)	1 2% (CAIDA data)
Ensemble of neural net model [411]	soft computing	RBPBoost system / Ensemble of neural net based classifiers	N	Yes	Yes	Yes	99 4% (DARPA 2000 DDoS data)	3 7% (DARPA 2000 DDoS data)
Autonomic mathematical model [424]	Machine learning	CPN based method	R	Yes	No	No	(ns2 simulation)	-

w-indicates real-time or non real-time
x-represents the scalability
y-possibility of unknown attack detection
z-possibility of dynamic signature updation
DR-Detection Rate and FAR-False Alarm Rate

8.5 System Modeling for DDoS Attack Detection

In this section, we attempt to model a system for detecting DDoS flooding attacks using an extended entropy metric We make the following assumptions.

- Routers have full control on in-and-out traffic flow.
- We collect packet and flow level traffic at the victim-end after various types of flooding attacks are launched.
- We sample network traffic in 5 minute intervals and during processing we further sample them in 10 seconds time intervals.

The proposed scheme is illustrated in Figure 8.13. This scheme is composed of mainly two parts, viz., DDoS attack detection using an extended entropy metric

[430] and an ensemble of extended entropy metrics. Extended entropy metric is defined below.

Definition 8.5.1. *Extended Entropy (EE) is defined as the sum of all entropies of parts of a system within a time interval.*

$$EE(x) = \sum_{i=1}^{T} EEM_i \tag{8.1}$$

where T is the time interval and EEM is the extended entropy metric of each part of a system.

In the detection scheme, we initially sample the network traffic into t number of intervals from a total time period $\mathbb{T}$. For each time interval, we compute the discrete probability distribution, packet intensity and individual entropies as discussed next. We compute both the local entropy metric difference between legitimate traffic and anomalous traffic, and the global entropy metric difference between legitimate traffic and anomalous traffic, if found greater than the local variation threshold ω_1 and less than the global variation threshold ω_2, then it marks the sample as attack otherwise normal. All attack samples can be used for the IP traceback purpose. We extend our scheme to use an ensemble of extended entropy metrics to improve the detection accuracy in near real-time.

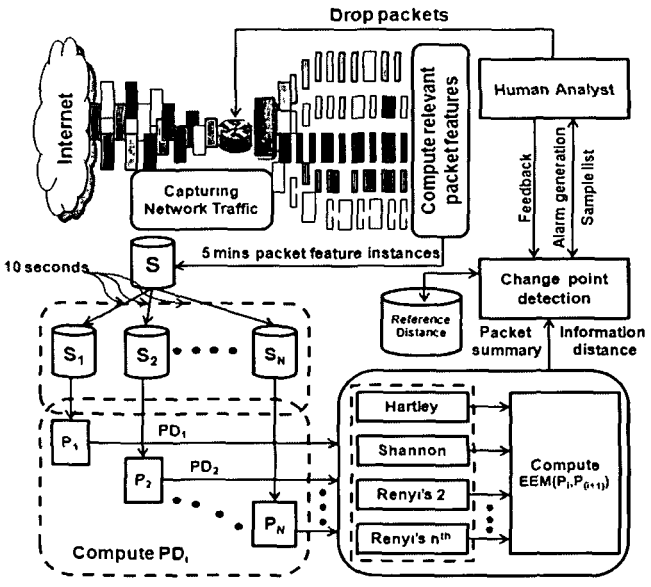


Figure 8.13: Concept of the proposed scheme

8.5.1 DDoS Attack Detection Scheme

In information theory, entropy is a measure of uncertainty in a random variable that forms the basis for distance and divergence measurements between probability densities. Larger values of entropy are expected when the information variable is more random. In contrast, the entropy value is expected to be small when the amount of uncertainty in the information variable is small [390] To quantify the randomness of a system, Renyi [431] introduced an entropy metric of order α as a mathematical generalization of Shannon entropy [426]. Let us consider a discrete probability distribution, $\mathbb{P} = p_1, p_2, p_3, \cdots p_n$, i e , $\sum_{i=1}^{n} p_i = 1$, $p_i \geq 0$. Then the Renyi's entropy of order α is defined as

$$H_\alpha(x) = \frac{1}{1-\alpha} log_2 \left(\sum_{i=1}^{n} p_i^\alpha \right) \tag{8.2}$$

where $\alpha \geq 0$, $\alpha \neq 1$. $p_i \geq 0$; if the values of p_i's are same, then the maximum entropy value is achieved, which is known as *Hartley entropy* [426]

$$H_0(x) = log_2 n \tag{8 3}$$

When $\alpha \to 1$, H_α converges to *Shannon entropy* [426].

$$H_1(x) = -\sum_{i=1}^{n} p_i log_2 p_i \tag{8.4}$$

If $\alpha = 2$, it is known as *collision entropy* or *Renyi's quadratic entropy* [431].

$$H_2(x) = -log_2 \sum_{i=1}^{n} p_i^2 \tag{8 5}$$

Finally, when $\alpha \to \infty$, $H_\infty(x)$ reaches the minimum information entropy value. Hence, we say that the generalization of information entropy is a non-increasing function of order α, i e., $H_{\alpha_1}(x) \geq H_{\alpha_2}(x)$, for $\alpha_1 < \alpha_2$, $\alpha > 0$. We define the probability and packet intensity computation as

$$p(x_i) = \frac{x_i}{\sum_{i=1}^{n} x_i} \tag{8.6}$$

$$f_i = \frac{\pi_i}{\sum_{j=1}^{N} \pi_j} \tag{8.7}$$

where $j = 1, 2, 3, \cdots N$, N represents the full time interval $\mathbb{T}$, π is the number of packets and n represents a smaller time interval t within $\mathbb{T}$ The Renyi's information entropy metric of order α can be rewritten as

$$EEM_{\alpha}(x) = \frac{t_i \times f_i}{N(1-\alpha)} log_2 \left(\sum_{i=1}^{n} p_i^{\alpha} \right) \tag{8.8}$$

where t_i is the time and f_i is the packet intensity for the i^{th} sample. We call this metric as the *extended entropy metric* (EEM) Based on this information entropy metric analysis, we consider different probability distributions for legitimate network traffic and attack traffic when detecting DDoS attacks. The flowchart of the proposed attack detection scheme is given in Figure 8.14

To support the proposed scheme, we introduce some definitions and lemmas below

Definition 8.5.2. ***DDoS flooding attack traffic*** *- Given a traffic sample S collected during a time interval T, a DDoS flooding attack traffic is a sub-sample, $A = \{a_1. a_2, a_3 \cdots a_S\}$ such that the difference of extended entropy metric between anomalous traffic and normal traffic is at least the minimum allowable threshold ω_1.*

Definition 8.5.3. ***Extended Entropy Metric (EEM)*** *- The extended entropy metric is simply the sum of entropy values of order α, used to rank each traffic sample within a time interval T for DDoS attack detection. The EEM metric value of attack traffic is higher than the EEM metric value of normal traffic within a time interval T.*

Definition 8.5.4. ***Locally anomalous traffic*** *- A DDoS flooding attack traffic sample is defined as locally anomalous iff $EEM(s_i - s_j) \geq \omega_1$ within time interval T, where s_i and s_j are anomalous and normal traffic respectively, and ω_1 is a user defined threshold.*

Definition 8.5.5. ***Globally anomalous traffic*** *- A DDoS flooding attack traffic sample is defined as globally anomalous iff $EEM(s_i - s_j) \leq \omega_2$ across two consecutive time interval t_i and t_{i+1} within a total time interval T, and ω_2 is a user defined threshold.*

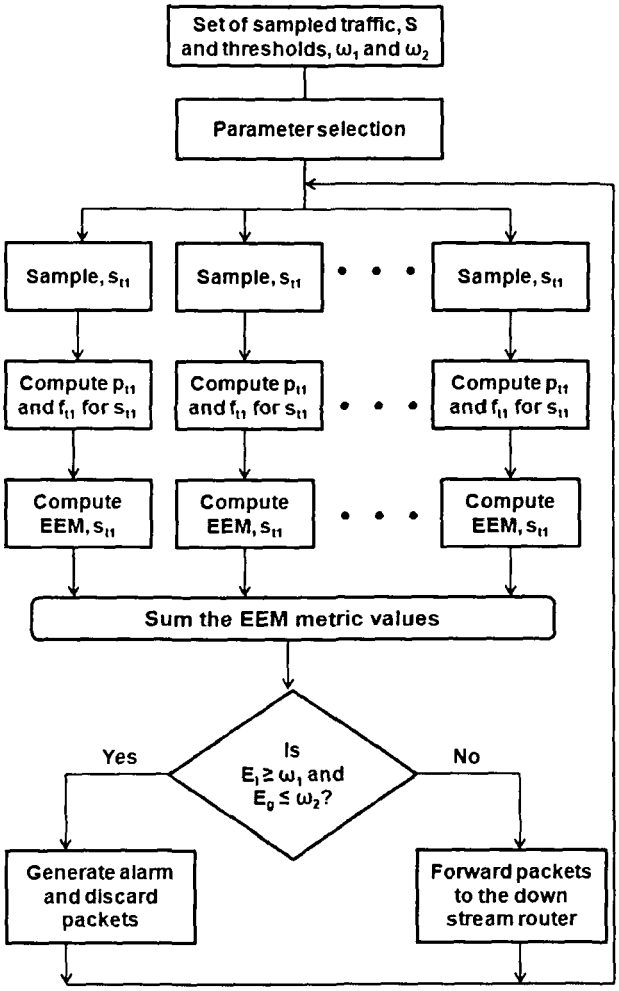


Figure 8.14: Flowchart of the proposed DDoS attack detection system

Lemma 8.5.6. *The maximum variation in DDoS flooding attack traffic sample* A, $A = \{a_1, a_2, a_3 \cdots a_S\}$ *in terms of EEM metric value is always less than the maximum variation for normal traffic.*

Proof. Let S_{ai} and S_{aj} be two samples of DDoS flooding attack traffic, and S_{ni} and S_{nj} be two samples of normal traffic. Based on [399] and according to *Definition* 8.5.2 and 8.5.3, the EEM metric value of attack traffic is higher than normal traffic, i.e., $EEM_{at} \gg EEM_{nt}$, where $EEM_{at} = EEM(S_{ai}) - EEM(S_{aj})$ and $EEM_{nt} = EEM(S_{ni}) - EEM(S_{nj})$ However, the flooding attack traffic is generated by using a program in other word, it is program controlled So. the variation among the traffic is ultimately limited within a bound. On the other hand, normal traffic variation has no such bound or control, and hence can be extended to great extent. So, the maximum variation of attack traffic in terms of EEM metric value is always less than the maximum variation for normal traffic. □

Lemma 8.5.7. *For a DDoS flooding attack traffic sample A, $A = \{a_1, a_2, a_3 \cdots a_S\}$ the EEM metric value is always larger than the Shannon entropy value.*

Proof. The proof of the above lemma is trivial from the representations of Shannon entropy and extended entropy metric given in Equation(8.4) and Equation(8.8) respectively. It is evident from the multiplying factor used in Equation(8.8). □

The DDoS Attack Detection Algorithm

The proposed information entropy metric-based DDoS flooding attack detection scheme attempts to detect four categories of DDoS flooding attacks as shown in Figure 8.3. In information theory, the value of Shannon entropy in a Gaussian distribution is higher than that of a Poisson distribution. The Renyi's generalized entropy value is lower than the Shannon entropy value when $\alpha > 1$. In contrast, the Renyi's generalized entropy value is higher than Shannon entropy when $0 \leq \alpha \leq 1$. But in case of the extended entropy metric, the EEM metric value is mostly greater than the Shannon entropy metric value. Hence, we can achieve better detection accuracy and lower false positive rate in the detection of all classes of DDoS flooding attacks. The steps of the proposed scheme are given in Algorithm 12.

The proposed scheme needs limited parameter estimation when detecting DDoS flooding attacks. A collaborative detection threshold that is needed can be estimated based on the spacing between legitimate traffic and anomalous traffic within the sampling period T for all classes of attacks.

Selective Ensemble of Extended Entropy Metric

An ensemble of extended entropy metrics is introduced mainly to improve the accuracy of the detection scheme. We combine weighted values of the extended entropy metric w.r.t the spacing between samples. We found good spacing between legitimate traffic and anomalous traffic when the order $\alpha > 1$. We define the ensemble of extended entropy metrics (EEEM) as follows.

Definition 8.5.8. ***EEEM metric*** *- The ensemble of extended entropy metric is defined as a metric to rank each traffic sample with a weighted sum of entropy values in a time interval T based on their entropy values of order $\alpha > 1$.*

Algorithm 12 The DDoS flooding attack detection algorithm

Input: network traffic $\mathbb{X}$ with respect to time window $\mathbb{T}$ and thresholds ω_1, and ω_2
Output: alarm information (attack or normal)

1: initialization: probability $p(x_i)$, packet intensity, f_i, and sample period, $T = 0$, where $i = 1, 2, 3, \cdots n$, $T = \{t_1, t_2, t_3, \cdots, t_N\}$, N is the full time interval.
2: sample the network traffic $\mathbb{X}$ received from upstream router R based on sampling period $\mathbb{T}$
3: compute probability distribution p_i and packet intensity f_i using Equation(8.6) and (8.7), respectively based on traffic features (i.e., sIP, dIP, packet size, etc.) for each sample within $\mathbb{T}$ sampling period of i^{th} sample.
4: compute extended entropy metric $H_\alpha(x)$ using Equation(8.8) for each sample within sampling period $\mathbb{T}$

$$S_i = \sum_{k=0}^{y} EEM_\alpha(s_{ik}) \tag{8.9}$$

$$S_j = \sum_{l=0}^{z} EEM_\alpha(s_{jl}) \tag{8.10}$$

$$E_{il} = |EEM_\alpha(s_i) - EEM_\alpha(s_j)| \tag{8.11}$$

$$E_{ig} = |EEM_\alpha(s_p) - EEM_\alpha(s_q)| \tag{8.12}$$

5: check against local variation threshold $E_{il} \geq \omega_1$ and global variation threshold $E_{ig} \leq \omega_2$, if so then generate alarm; otherwise, router forward the packet to the downstream routers.
6: go to step 2.

$$EEEM(X) = \sum_{i=1}^{m} \sum_{j=1}^{n} \frac{1}{W_i} \times EEM_j(X) \tag{8.13}$$

where W is the weight and $EEM_j(X)$ is the extended metric value. The weight W is defined as

$$W_i = \begin{cases} S_k & \text{if } EEM(x) \geq \sigma \\ 1 & \text{otherwise} \end{cases}$$

where σ is the threshold for selection of the weight value iff $\alpha > 1$. We use ω_3 as the threshold for attack detection in selective ensemble of extended entropy metrics.

8.5.2 Complexity Analysis

The detection scheme takes $O(Tn)$ time during detection, where T is the time interval and n is the number of instances within a sample The time complexity for the detection scheme is linear w r t the size of the dataset and the number of features Hence, our scheme is computationally efficient in detecting DDoS flooding attacks with low false alarm rate and time

8.6 Performance Evaluation

Performance evaluation is important for any DDoS attack defense system Performance evaluation is highly dependent on (i) the approach, (ii) deployment status and (iii) whether it is possible to dynamically update profiles When designing a DDoS attack defense scheme, these issues should be taken into consideration There are many tools available to launch DDoS attacks in the literature [384, 432] The architectures are almost always the same Some are made by attackers by slightly modifying others Table 8 7 presents some of the tools with brief descriptions

In our experiments, three different datasets viz , MIT Lincoln Laboratory [446], CAIDA DDoS 2007 [252] and TUIDS DDoS[1] datasets are used to detect four classes of DDoS flooding attacks as discussed above The TUIDS DDoS datasets is prepared using our testbed described in Chapter 4 The architecture of the TUIDS testbed with demilitarized zone (DMZ) is shown again in Figure 8 15 The testbed is composed of 5 different networks inside the Tezpur University campus The hosts are divided into several VLANs, each VLAN belonging to an L3 switch or an L2 switch inside the network The attackers are placed in both wired and wireless networks with reflectors but the target is placed inside the internal network

8.6.1 Datasets

The MIT Lincoln Laboratory tcpdump data is used as real-time normal network traffic the data does not contain any attacks (see Figure 8 16 for a depiction of the normal traffic scenario from the MIT Laboratory) The CAIDA DDoS 2007 dataset

[1]http //agnigarh tezu ernet in/~dkb/resource html

Table 8.7: DDoS tools and description

Name and Ref	Description	Protocol	Attack
Trinoo [433, 434]	(i) Widely used by attackers as well as research community (ii) A bandwidth depletion attack tool, used to launch coordinated UDP flood attacks against one or many IP addresses (iii) Fixed size UDP packets are sent to the victim machine's random ports (iv) Does not spoof source addresses (v) Implements UDP Flood attacks against the target victim	UDP	UDP flood
Tribe Flood Network (TFN) [435]	(i) Able to wage both bandwidth depletion and resource depletion attacks (ii) Uses a command line interface to communicate between the attacker and the control master program (ii) Offers no encryption between agents and handlers or between handlers and the attacker (iii) Allows TCP SYN and ICMP flood as well as smurf attacks	UDP, ICMP TCP	TCP SYN flood, ICMP flood, smurf
TFN2K [436]	(i) Developed using the TFN DDoS attack tool (ii) Adds encrypted messaging among all of the attack components [437] (iii) Communications between real attacker and control master program are encrypted using a key-based CAST-256 algorithm [438] (iv) Conducts covert exercises to hide itself from intrusion detection systems (v) Can forge packets that appear to come from neighboring machines (vi) Provides other options such as TARGA and MIX attack [439]	TCP, UDP, ICMP	smurf, SYN flood, UDP flood, ICMP flood
Stacheldraht [440]	(i) Based on early versions of TFN, it eliminates some weak points by combining features of Trinoo (ii) Performs updates on the agents automatically (iii) Provides a secure telnet connection via symmetric key encryption among the attackers and handlers (iv) Communicates through TCP and ICMP packets	TCP, UDP, ICMP	TCP SYN flood, UDP flood, ICMP echo request flood
mstream [441]	(i) Uses spoofed TCP packets with the ACK flag set to attack the target (ii) A simple point-to-point TCP ACK flooding tool to overwhelm the tables used by fast routing routines in switches (iii) Communications are not encrypted, and performed through TCP/UDP packets, zombie is connected via telnet by master (iv) Target gets hit by ACK packets and sends TCP RST to non-existent IP addresses (v) Routers return 'ICMP unreachable causing more bandwidth starvation (vi) Possesses very limited control features and can spoof by randomizing all 32 bits of the source IP address	TCP, UDP	TCP ACK flood
Shaft [442]	(i) A successor of Trinoo (ii) Uses UDP communication between handlers and agents (iii) Shaft provides UDP/ICMP/TCP flooding attack options it randomizes source IP address and source port in packets (iv) The size of packets remains fixed during the attack (v) Able to switch the handler's IP address and port in real time during the attack (vii) Able to switch control master servers and ports in real time, hence making detection by intrusion detection tools difficult	TCP, UDP, ICMP	TCP/UDP/ ICMP flood
Trinity v3 [443]	(i) Various TCP floods are used by randomizing all 32 bits of the source IP address TCP fragment floods, TCP established floods, TCP RST packet floods and TCP random flag packet floods (ii) Generates TCP flood packets with random control flags set to provide a wider set of TCP based attacks	TCP, UDP	TCP fragment floods TCP RST packet floods TCP random flag packet floods, TCP established floods
Knight [444]	(i) A very lightweight yet powerful IRC based attack tool (ii) Provides SYN attacks, UDP Flood attacks, and an urgent pointer flooder [445] (iii) Designed to run on Windows operating systems and has features such as an automatic updater via http or ftp, a checksum generator and more (iv) Uses Trojan horse program called Back Orifice for installation in the target host	TCP, UDP	UDP, TCP flood, SYN and PUSH+ACH flood
LOIC [377]	(i) A powerful anonymous attacking tool via IRC (ii) Operates in three methods of attack TCP UDP and HTTP (iii) Exists in two versions binary version and web-based version	TCP, UDP, HTTP	UDP, TCP HTTP flood

uses real-time DDoS attack data with four classes of attack scenarios, viz , constant rate, increasing rate, pulsing rate and subgroup attack (see Figures 8 17, 8 18, 8 19 8 20 for classes of DDoS attack scenarios) The CAIDA dataset contains 5 minutes

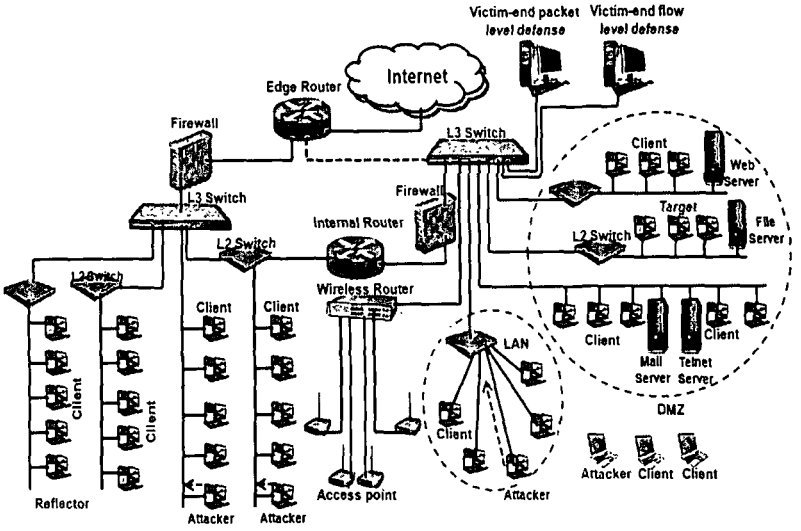


Figure 8.15: TUIDS testbed network architecture with DMZ

(i.e., 300 seconds) of anonymized traffic from a DDoS attack on August 4, 2007. This trace includes only attack traffic to the victim and responses from the victim; nonattack attack traffic has been removed as much as possible. Finally, the TUIDS DDoS dataset also contains different classes of attacks scenarios like CAIDA. The TUIDS DDoS dataset contains six different attacks for generation and analysis of near real-time DDoS attack detection. The list of attacks and generation tools used by the TUIDS DDoS dataset is given in Table 8.8. We have chosen six most powerful flooding attacks and launch them in the testbed. As the most attackers use three different types of protocols (i.e., TCP, UDP, and ICMP) during sending their malicious traffic to the target host or network. Hence, we use TCP, UDP and ICMP protocol-based flooding attack during testing our proposed method in near real-time in our testbed. These attacks are generated using openly available standard DDoS attack generation tools.

Table 8.8: List of real-life attacks and their generation tools

Attack name	Generation tool
1.syn-flood	LOIC
2.rst-flood	Trinity v3
3.udp-flood	LOIC
4.ping-flood	DDoS ping v2.0
5.fraggle udp-flood	Trinoo
6.smurf icmp-flood	TFN2K

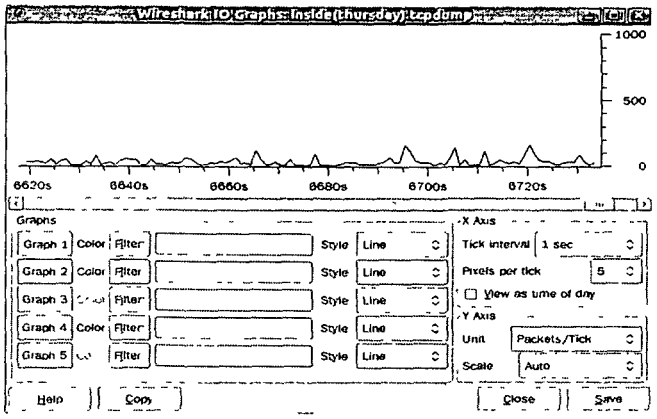


Figure 8.16: Normal (attack free) traffic scenario from MIT Lincoln Laboratory data. X-axis denotes intervals (seconds) and Y-axis denotes packets/tick (unit).

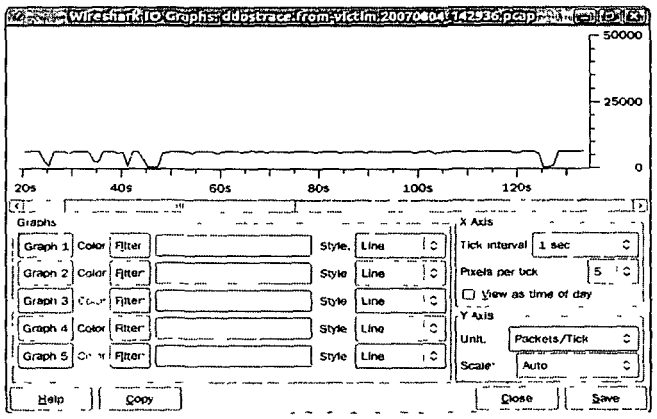


Figure 8.17: DDoS attack scenarios from CAIDA: constant rate attack. X-axis denotes intervals (seconds) and Y-axis denotes packets/tick (unit).

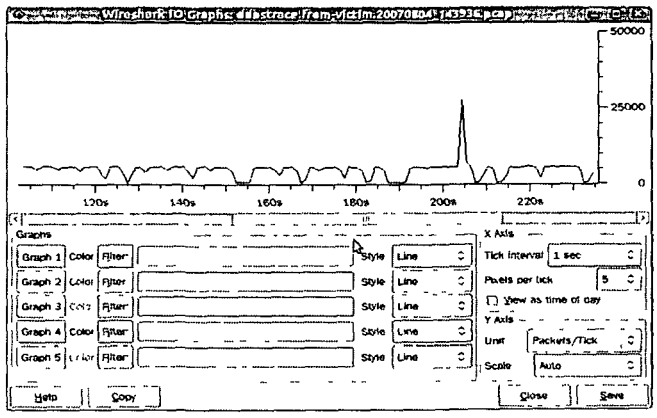


Figure 8.18: DDoS attack scenarios from CAIDA: pulsing rate attack. X-axis denotes intervals (seconds) and Y-axis denotes packets/tick (unit).

8.6.2 Experimental Results

To evaluate the performance of the proposed method, we initially sample the network traffic into 10 second windows for each dataset. We identify the static IP

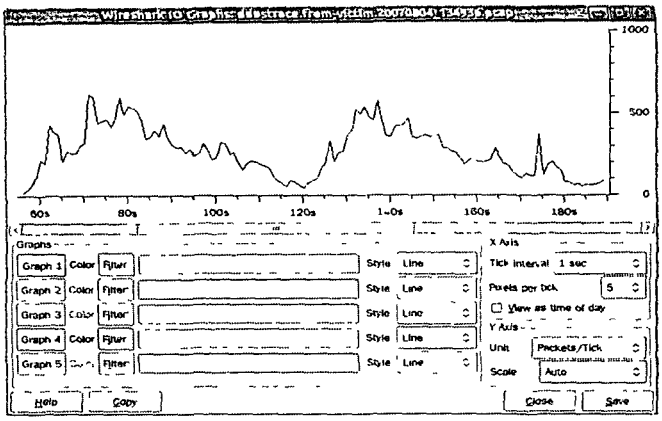


Figure 8.19: DDoS attack scenarios from CAIDA: increasing rate attack. X-axis denotes intervals (seconds) and Y-axis denotes packets/tick (unit).

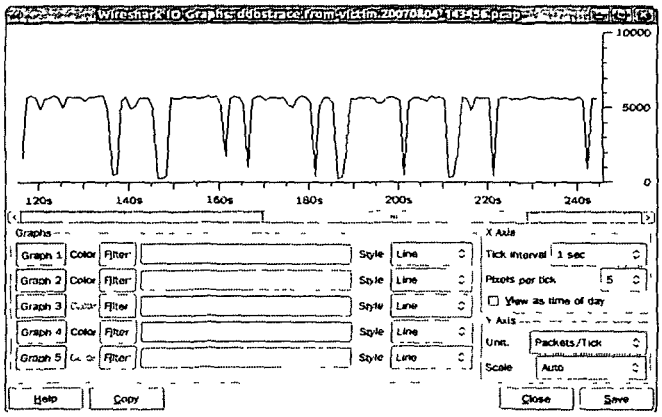


Figure 8.20: DDoS attack scenarios from CAIDA: subgroup attack. X-axis denotes intervals (seconds) and Y-axis denotes packets/tick (unit).

packets and compute discrete probability distribution for each sample. The probability of IP packet distribution in three scenarios: (a) attack traffic, (b) normal traffic, and (c) mixed traffic (contains both normal and attack traffic) are shown in Figures 8.21, 8.22, 8.23, respectively.

We compute entropy using the extended entropy metric for each probability distribution and average them for each sample. To test our proposed scheme, we compute the extended entropy metric of different orders using α and compare with Shannon entropy within a sampled period of legitimate traffic and anomalous traffic. Figure 8.24 presents the value of Shannon entropy and the extended entropy metric for different orders α. We consider order $\alpha = 0$ to 15 and vary the spacing between legitimate traffic and attack traffic. It demonstrates that the proposed scheme outperforms the use of Shannon entropy, especially in detecting DDoS flooding

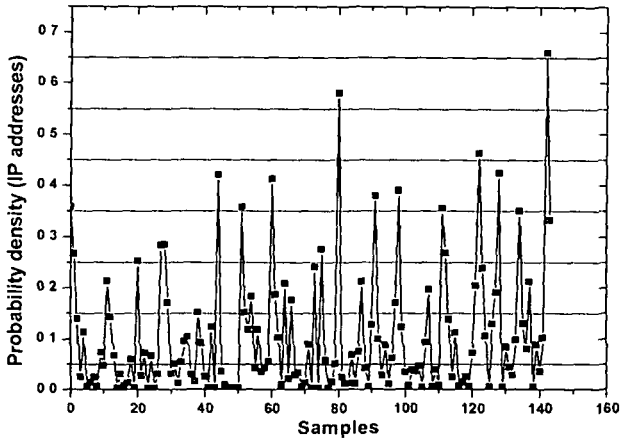


Figure 8.21: Probability distribution of IP addresses in normal traffic

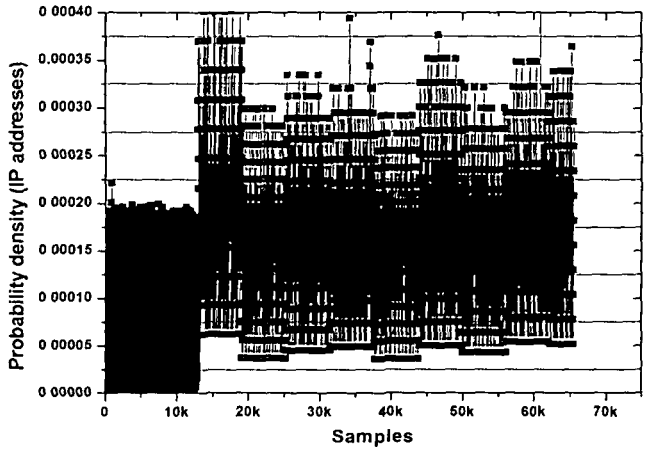


Figure 8.22: Probability distribution of IP addresses in attack traffic

attacks because it can obtain significant spacing between legitimate traffic and attack traffic. It also shows that the extended entropy metric values gradually increase along with the order α w.r.t. the traffic rate, which is almost linear. To test our scheme globally, we test for each attack class discussed above. The results are given in Figures 8.25, 8.26, 8.27, 8.28 for CAIDA dataset and Figures 8.29 and 8.30 for the TUIDS datasets. The proposed scheme performs very well in detecting DDoS flooding attacks including DDoS attacks generated in our testbed.

We further compute the detection rate and false positive rate based on the

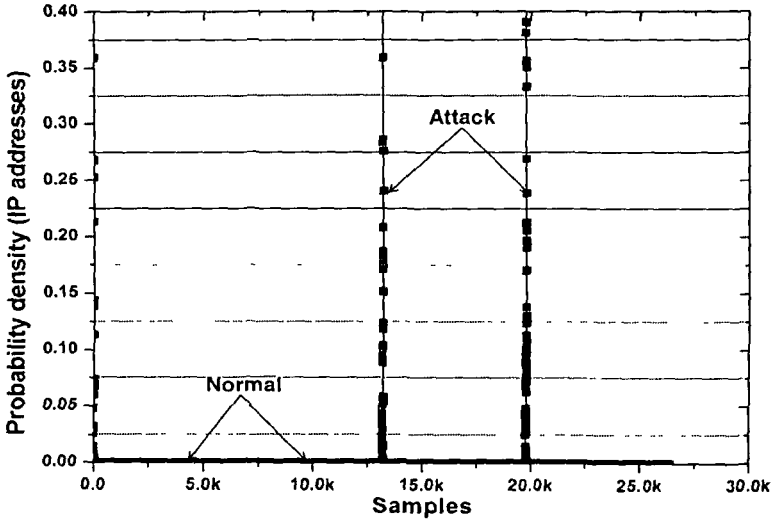


Figure 8.23: Probability distribution of IP addresses in mixed traffic (contains both normal and attack traffic)

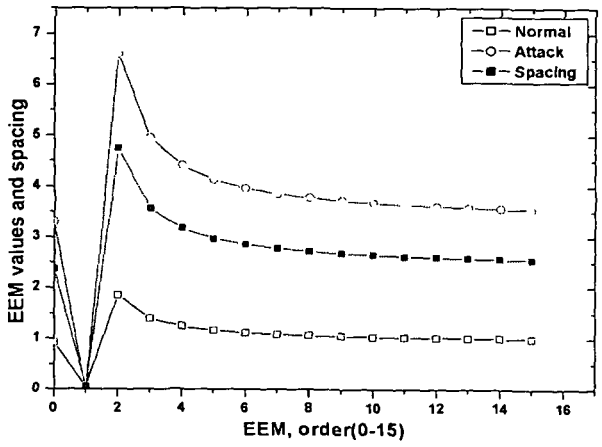


Figure 8.24: EEM metric values for normal and attack traffic with spacing in between

samples within a time interval. The results of our extended entropy metric and ensemble of EEM in comparison to Shannon entropy is given in Figure 8.31. Our detection scheme can effectively detect DDoS attacks with the least amount of false alarms and performs well in comparison to several competing algorithms [379, 399, 410, 428, 447].

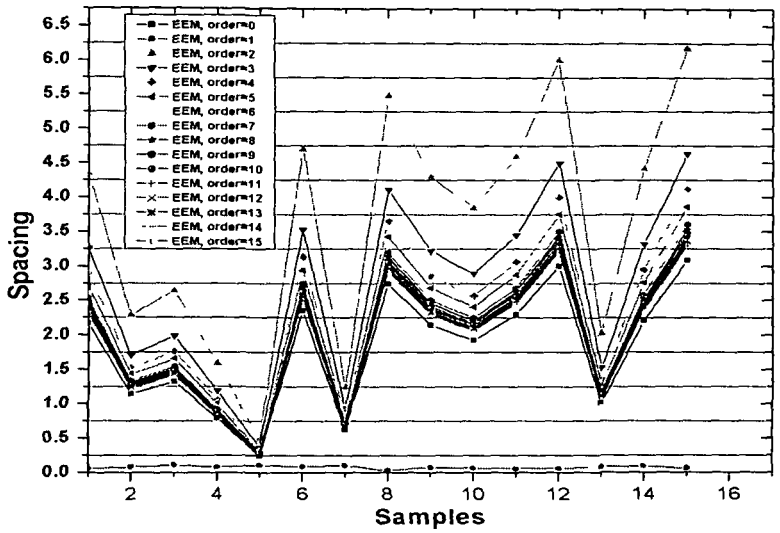


Figure 8.25: CAIDA dataset: spacing between legitimate and anomalous traffic in constant rate traffic

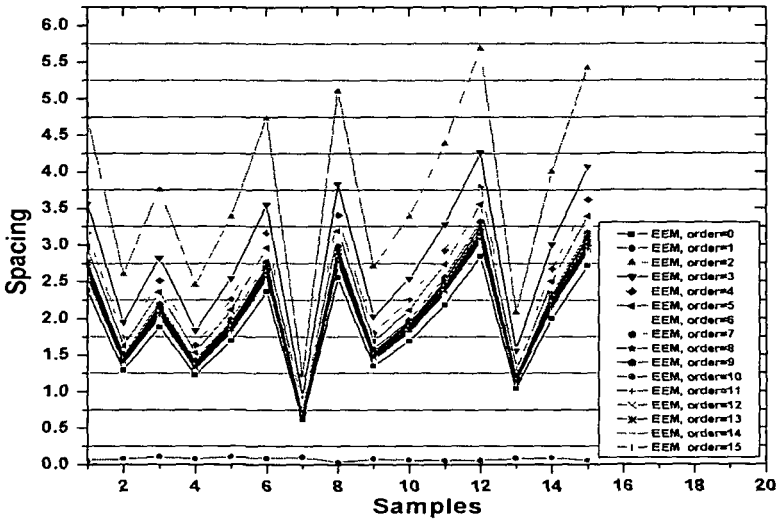


Figure 8.26: CAIDA dataset: spacing between legitimate and anomalous traffic in pulsing rate traffic

Size of Split Window Analysis

The size of monitoring window is decided based on the time taken for analysis of traffic Throughout our experiment, we set the optimal split window size as $t = 10$ seconds, i.e., sub-sample window size The total time $T = 300$ seconds is taken for each sample traffic during analysis. Due to huge amount of traffic, we took minimum split window size for analysis and can detect DDoS attacks effectively in less time.

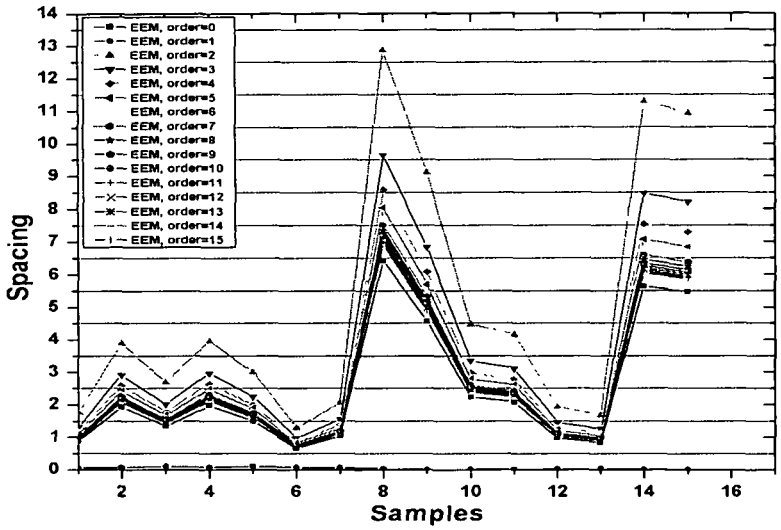


Figure 8.27: CAIDA dataset: spacing between legitimate and anomalous traffic in increasing rate traffic

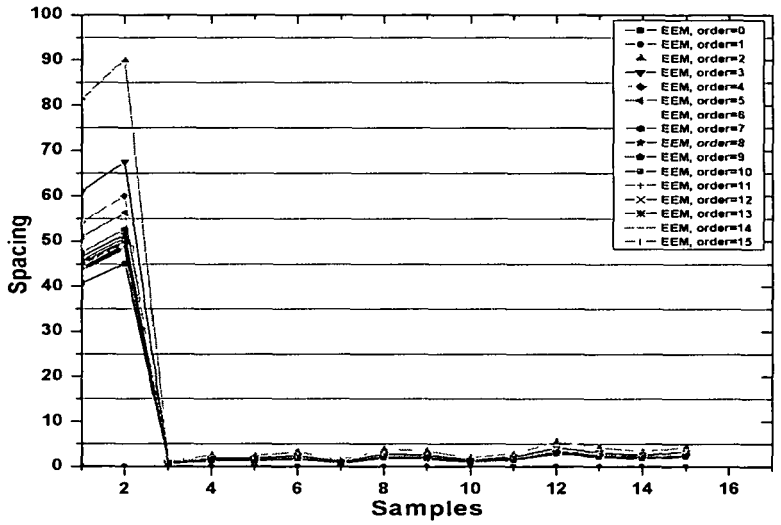


Figure 8.28: CAIDA dataset: spacing between legitimate and anomalous traffic in subgroup attack traffic

Selection of Minimum Features

Normally, there are several traffic features considered during network attack detection and requires more time to detect attacks. Therefore, we used three optimal parameters such as source IP, destination IP and protocol to improve detection rate significantly. These parameters is selected from the raw network traffic in enterprize network.

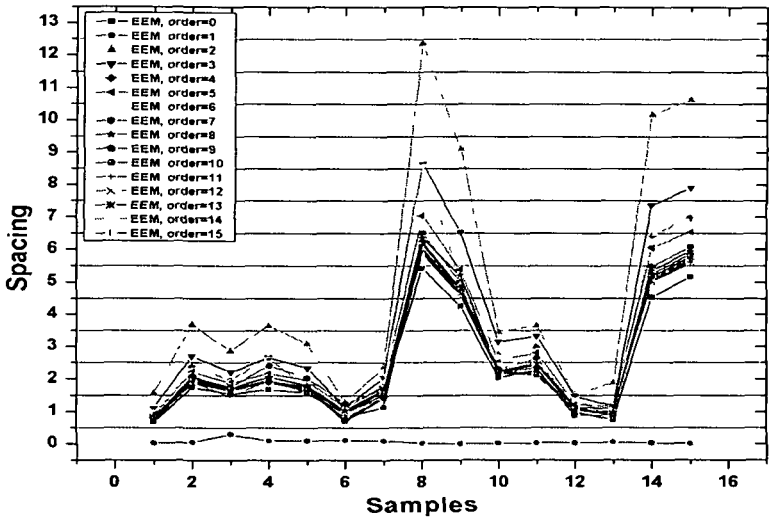


Figure 8.29: TUIDS dataset: spacing between legitimate and anomalous traffic in packet level traffic

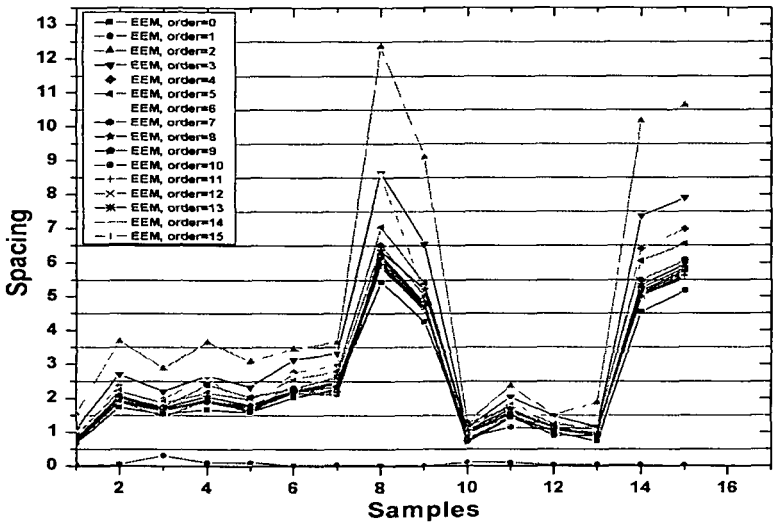


Figure 8.30: TUIDS dataset: spacing between legitimate and anomalous traffic in flow level traffic

Threshold $(\omega_1, \omega_2, \omega_3)$ Analysis

In order to estimate the threshold values, we consider heuristic approach for each threshold value identification In our experiment, we used three thresholds for different levels such as ω_1, ω_2 and ω_3, where ω_1 is the threshold for local variations, ω_2 is the threshold for global variations and ω_3 is the threshold for the selective ensemble of extended entropy metrics for DDoS detection We obtain better results while $\omega_1 >= 0.0280$, $\omega_2 <= 15.6818$ and $\omega_3 >= 3\ 4301$ in CAIDA DDoS dataset.

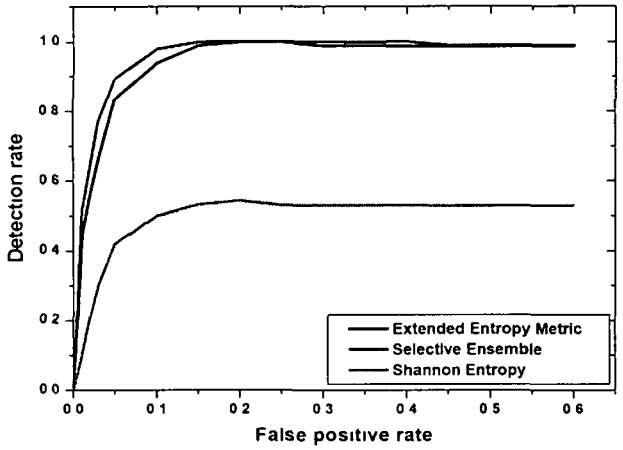


Figure 8.31: ROC of extended entropy metric and selective ensemble in comparison to Shannon entropy

But in case of TUIDS DDoS dataset $\omega_1 >= 0\ 01935$ $\omega_2 <= 11\ 2538$ and $\omega_3 >= 2\ 7184$

Information Entropy Analysis

In information entropy analysis we have computed information entropy of order $\alpha = 0, 1\ 2,\ \ 15$ during our experiment However, we obtain maximum extended entropy metric value in order $\alpha = 2$ and minimum in order $\alpha = 1$, i e , Shannon entropy So the difference between normal and attack traffic gets higher than that the difference in Shannon entropy Also the global difference of EEM value between two consecutive sub-sample is higher than that the difference in Shannon entropy

Peak Analysis

In peak analysis, we basically consider the maximum difference of extended entropy metric value for the different attacks including (a) constant rate, (b) pulsing rate, (c) increasing rate and (d) subgroup attack in both CAIDA and TUIDS DDoS datasets We obtain the peak value in case of CAIDA datasets (a) constant rate, peak value = 6 26, (b) pulsing rate, peak value = 5 74, (c) increasing rate, peak value = 12 72 and (d) subgroup attacks, peak value = 90 01 (see Figures 8 25, 8 26, 8 27 and 8 28) In TUIDS dataset, we obtain the peak value in (a) packet level,

peak value = 12.48 and (b) flow level, peak value = 12 51 (see Figures 8 29 and 8.30).

8.6.3 Discussion

To detect DDoS flooding attacks, it is useful to detect with a minimum number of IP traffic features Many detection schemes use the distribution of either IP addresses or IP packet sizes. The IP address-based method uses IP addresses and computes the information entropy metric by computing the probability of each unique IP addresses within a certain time interval. A bigger entropy value represents higher randomness in the IP addresses Based on the distribution of IP addresses, it estimates the change in information entropy metric difference between legitimate traffic and anomalous traffic. We analyze our scheme using several real-life DDoS attack datasets Our scheme has the following distinguishing features.

- The scheme can detect anomalous traffic containing DDoS flooding attacks with low false alarm rate and low time complexity.
- The detection scheme uses a minimum number of IP traffic features during attack detection, which makes it more cost-effective
- The detection scheme is dependent on a minimum number of parameters.

8.7 Summary

In the beginning of this chapter, we presented an overview of DDoS attacks, a taxonomy, detection methods and tools. Then, we proposed an effective information entropy metric known as the extended entropy metric that can detect DDoS attacks with several attack scenarios. The experimental results demonstrate that the proposed scheme works effectively and stably in detecting DDoS attack traffic. It magnifies the spacing between the legitimate traffic and attack traffic significantly, which makes it easier to detect DDoS attacks. It also reduces the false alarm rate significantly when detecting DDoS attacks. In addition, we extended our work to use an ensemble of extended entropy metrics for increasing detection rate in near

real-time. The use of the ensemble of extended metrics also produces better results during detection of DDoS attacks. We also show that our proposed extended entropy metric-based DDoS attack detection scheme performs well in comparison to traditional detection schemes.

Chapter 9

Conclusions and Future Directions

This chapter summarizes the thesis with discussion of (a) the findings and the contributions to the state-of-the-art in the disciplines covered by this work, and (b) future work, those directions that our research will undertake addressing open issues that deserve further attention.

9.1 Summary of Research Contributions

With the explosive growth of the Internet during last two decades, Internet-based attacks on large scale enterprise networks have grown rapidly. It is important to keep secure enterprise networks from these threats The main motivation of the research conducted during this thesis is to monitor and analyze network traffic for anomaly detection.

This thesis has mainly focused on applying data mining techniques in network traffic monitoring and analysis to address the problem of efficient anomaly detection and has evaluated their performance using real world benchmark network intrusion datasets. Summarized, the most important contributions of this thesis are the following.

- In Chapter 2, we present the anomalies in networks, taxonomy of network based attacks, anomaly detection, and evaluation criteria. This chapter also discusses definitions, causes, sources, types of anomalies in networks or hosts and detection approaches for anomaly detection with generic architecture for each of them.

- A structured and comprehensive review on network traffic anomaly detection, methods, systems and tools has been reported in Chapter 3. It includes detection methods and systems under six different categories, viz , statistical, classification, clustering and outlier-based, soft computing, knowledge-based, and combination learners. We also list common and relevant tools used by both attackers and network defenders during live network traffic capture, visualization and analysis

- In Chapter 4, we present a systematic approach towards generation of benchmark network intrusion datasets Three separate datasets are prepared using the TUIDS testbed architecture. They are (i) TUIDS intrusion dataset, (ii) TUIDS coordinated scan dataset, and (iii) TUIDS DDoS dataset Out of several real world attacks, we have chosen and incorporated 28 attacks in preparing our datasets. We have been able to provide a path and a template that ultimately leads to generate a dataset that reflects the appropriate amounts of normality, anomalousness as well as realism Our datasets demonstrate several features including that (i) they are built by incorporation of latest network scenarios with real network traffic, (ii) We extraction maximum amounts of packet and flow level features during generation of final datasets, and (iii) They are large in size to support effective validation of the performance of detection method.

- In Chapter 5 we initially examine the state-of-the-art of modern port scans and detection methods. Next we introduce an adaptive outlier-based method for coordinated scan detection In contrast to exploring features for clustering and visualization, AOCD uses random sample selection using a linear congruential generator for distinct profile generation. We also propose an outlier score function to test each candidate object to identify coordinated port scans using score values The method reports each candidate object as normal or coordinated port scan with respect to a user defined threshold It is capable of detecting coordinated scans that have a stealthy and horizontal or strobe footprint across a contiguous network address space. We test this method using several real world datasets including the TUIDS coordinated

scan dataset and KDDcup99 probe dataset. Due to non availability of similar datasets from other sources, we use the KDDcup99 probe dataset to evaluate the method. Coordinated scans are performed in an isolated environment, combining network traffic traces with those collected from live networks. This method achieves high detection accuracy and low false positive rate on various real life datasets in comparison to existing coordinated scan detection methods.

- In Chapter 6, we introduce an effective outlier detection technique to identify anomalies in high dimensional network traffic datasets. It introduces a tree-based subspace clustering technique to cluster large high dimensional datasets The clustering technique arranges the data in depth-first manner before applying our algorithm for network anomaly detection It also uses the outlier score function to rank each candidate data object based on scores. The key features of this technique are (i) It is able to successfully detect all outlier cases that we consider and (ii) It can use any proximity measure for the computation of anomaly score. But choosing the threshold value is a difficult task during network traffic anomaly detection. We choose this threshold based on a heuristic approach. The performance of the proposed technique is assessed using several datasets, viz , (i) synthetic, (ii) UCI ML repository datasets, (iii) TUIDS intrusion dataset, (iv) TUIDS coordinated scan dataset, and (v) KDDcup99 and NSL-KDD datasets. Our technique performs well compared to similar algorithms.

- Chapter 7 presents a tree-based subspace clustering technique for unsupervised network anomaly detection in high dimensional large datasets. It generates the approximate number of clusters without having any prior knowledge of the domain. We analyzed cluster stability for each cluster by using an ensemble of cluster indices. We also introduce a multi-objective cluster labelling technique to label each stable cluster as normal or anomalous The major attractions of this method are: (i) TreeCLUSS does not require the number of clusters apriori, (ii) It is free from the restriction of using a specific proximity measure, (iii) CLUSSLab is a multi-objective cluster labelling technique

containing an effective unsupervised feature clustering technique to identify a dominant feature subset for each cluster, and (iv) TreeCLUSS exhibits a high detection rate and a low false positive rate, especially in case of probe, U2R, and R2L attacks Thus, we are able to establish the proposed method to be superior compared to competing network anomaly detection techniques. We also demonstrate that the results produced by our method are statistically significant.

- Finally, in Chapter 8, we discuss an overview of DDoS attacks, generic architectures, detection schemes and tools. We use information entropy metrics for DDoS flooding attack detection. We propose an extended entropy metric-based victim-end scheme for detecting classes of DDoS flooding attacks by measuring the difference of the metric between legitimate traffic and attack traffic. The method exploits a generalized entropy metric with packet intensity computation over sampled traffic within a time interval. We also extend the mechanism to an ensemble of extended entropy metrics for increasing detection rate in near real-time. The proposed scheme is evaluated using several real world DDoS datasets and it outperforms competing schemes when detecting classes of DDoS flooding attacks, viz , constant rate, increasing rate, pulsing rate and subgroup attack.

9.2 Future Work

Despite being well-investigated fields, the topics covered in this thesis are far from being dead-ends. This final section is devoted to discuss possible continuations for the research carried out in this thesis, some being part of our ongoing work.

- Even though several network intrusion datasets are available for the research community, they lack comprehensiveness and completeness, and are not available in the public domain Therefore, we provide a template toward generation and preparation of benchmark network intrusion datasets. It is possible to further extend the work by incorporating both low rate and high rate attacks for all categories of datasets

- Coordinated scan represent a community effort to reduce network bandwidth when attempting to quickly gain the vulnerability information, helpful to attackers We introduce an adaptive outlier-based technique to detect coordinated scans. But its observations of scan activities are limited to the network layer. So, it needs to be extended further to detect address resolution protocol (ARP) based coordinated scanning, low-rate coordinated scans, and high rate coordinated scans within stipulated time periods.

- An outlier-based network anomaly detection technique can play an important role in identifying types of attacks We develop a distance-based outlier detection technique and apply it to anomaly detection. There is ample scope still to develop a parameter free hybrid outlier detection technique for mixed type data to efficiently detect a larger number of attacks that combine both distance and density features.

- If a method can detect network traffic anomalies without using any domain knowledge, it is known as an unsupervised method. Such methods always generate large amounts of false alarms because they do not use appropriately labeled data for training We introduce a completely unsupervised network anomaly detection method. Developing a real-time unsupervised network anomaly detection method for mixed-type data remains a challenging task.

- Though we develop an information metric-based scheme to detect DDoS flooding attacks, there are several open challenges to achieve real-time performance. Hence we are planning to apply our outlier-based technique to detect DDoS flooding attacks and also aim to extend our scheme with an effective IP traceback mechanism. Finally, we note that the development of low-rate DDoS attack detection with an appropriate IP traceback technique is another open problem

Bibliography

[1] Breunig, M. M. *et al* LOF: Identifying Density-based Local Outliers. In *Proc. of the ACM SIGMOD International Conference on Management of data.* 386–395, 2000.

[2] Bay, S D & Schwabacher, M Mining Distance-based Outliers in Near Linear Time with Randomization and a Simple Pruning Rule In *Proc. of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* 29–38, ACM, 2003

[3] Pei, Y. *et al.* An Efficient Reference-Based Approach to Outlier Detection in Large Datasets In *Proc of the 6th International Conference on Data Mining.* 478–487, IEEE, USA, 2006

[4] Horng, S. J. *et al* A Novel Intrusion Detection System Based on Hierarchical Clustering and Support Vector Machines *Expert Systems with Applications: An International Journal* **38** (1), 306–313, 2011.

[5] Bhuyan, M. H. *et al* Surveying Port Scans and Their Detection Methodologies. *Comp. J.* **54** (10), 1565–1581, 2011.

[6] Beghdad, R. Critical Study of Supervised Learning Techniques in Predicting Attacks. *Information Security Journal: A Global Perspective* **19** (1), 22–35, 2010

[7] Jiang, S. *et al.* A Clustering-based Method for Unsupervised Intrusion Detections. *Pattern Recognition Letters* **27** (7) 802–810, 2006

[8] Tsai, C.-F. & Lin, C.-Y. A Triangle Area based Nearest Neighbors Approach to Intrusion Detection *Pattern Recognition* **43** (1), 222–229, 2010.

[9] Prolexic. Prolexic Quarterly Global DDoS Attack Report, Q1 2013. http://www prolexic.com/, 2013. USA.

[10] Bhuyan, M. H *et al.* Detecting Distributed Denial of Service Attacks. Methods, Tools and Future Directions. *The Computer Journal, DOI· 10.1093/comjnl/bxt031, 2013* , 2013

[11] Patcha, A. & Park, J. M. An Overview of Anomaly Detection Techniques· Existing Solutions and Latest Technological Trends. *Computer Networks* **51** (12), 3448–3470, 2007

[12] Chandola, V. *et al* Anomaly Detection: A Survey. *ACM Computing Surveys* **41** (3), 1–58, 2009.

[13] Wu, S. X. & Banzhaf, W The Use of Computational Intelligence in Intrusion Detection Systems A Review *Applied Soft Computing* **10** (1), 1–35, 2010

[14] Fayyad, U. *et al* The KDD Process for Extracting Useful Knowledge from Volumes of Data *Communication of the ACM* **39** (11), 27–34, 1996.

[15] Tan, P. N. *et al. Introduction to Data Mining*, Addison-Wesley, 2009, fourth edition edn.

[16] Tanenbaum, A *Computer Networks*, Prentice Hall Professional Technical Reference, 2002, 4th edn.

[17] Wood, P. *et al.* Internet Security Threat Report. Tech. Rep. 17, Symantec, USA, 2012.

[18] Thottan, M. & Ji, C. Anomaly Detection in IP Networks. *IEEE Transactions on Signal Processing* **51** (8), 2191–2204, 2003

[19] Kumar, V. Parallel and Distributed Computing for Cybersecurity. *IEEE Distributed Systems Online* **6** (10), 2005.

[20] Bace, R. & Mell, P. Intrusion Detection Systems. Tech. Rep. SP800-31, NIST Special Publications, US Department of Defence, USA, 2001

[21] Su M -Y Using Clustering to Improve the KNN-based Classifiers for Online Anomaly Network Traffic Identification *Journal of Network and Computer Applications* **34** (2) 722–730, 2011

[22] 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** (10), 2201–2212 2007

[23] Laskov, P *et al* Incremental Support Vector Learning Analysis, Implementation and Applications *Journal of Machine Learning Research* **7**, 1909–1936 2006

[24] Yi, Y *et al* Incremental SVM Based on Reserved Set for Network Intrusion Detection *Expert Systems with Applications* **38** (6), 7698–7707, 2011

[25] Khreich W *et al* Adaptive ROC-based Ensembles of HMMs Applied to Anomaly Detection *Pattern Recognition* **45** (1), 208–230 2012

[26] Burbeck, K & Nadjm-Tehrani, S Adaptive Real-time Anomaly Detection with Incremental Clustering *Information Security Technical Report* **12** (1), 56–67, 2007

[27] Erman J *et al* Semi-supervised Network Traffic Classification *SIGMETRICS Perform Eval Rev* **35** (1) 369–370, 2007

[28] Zhang J *et al* Semi-supervised and Compound Classification of Network Traffic In *Proc of the 32nd International Conference on Distributed Computing Systems Workshops* 617–621, 2012

[29] Song, J *et al* Toward a More Practical Unsupervised Anomaly Detection System *Information Sciences* **231** 2011 URL `http //dx doi org/10 1016/j ins 2011 08 011`

[30] Noto K *et al* FRaC A Feature-modeling Approach for Semi-supervised and Unsupervised Anomaly Detection *Data Min Knowl Discov* **25** (1), 109–133, 2012

[31] Casas, P *et al* Unsupervised Network Intrusion Detection Systems Detecting the Unknown without Knowledge *Computer Communication* **35** (7), 772–783, 2012

[32] Zhang J & Zulkernine, M A Hybrid Network Intrusion Detection Technique Using Random Forests In *Proc of the 1st International Conference on Availability, Reliability and Security* 262–269, IEEE CS, USA 2006

[33] Aydin M A *et al* A Hybrid Intrusion Detection System Design for Computer Network Security *Computers & Electrical Engineering* **35** (3), 517–526, 2009

[34] Panda, M *et al* Hybrid Intelligent Systems for Detecting Network Intrusions *Computer Physics Communications* **Early**, 2012

[35] Anderson, J Computer Security Threat Monitoring and Surveillance Tech Rep 215 646-4706 James P Anderson Co , Fort Washington, Pennsylvania, 1980

[36] Kizza, J M *Computer Network Security*, Springer 2005, 1st edn

[37] Chandola, V *et al* Anomaly Detection A Survey *ACM Computing Surveys* **41** (3), 15 1–15 58, 2009

[38] Hunt, R & Hansman S A Taxonomy of Network and Computer Attack Methodologies 2003

[39] Hansman, S & Hunt R A Taxonomy of Network and Computer Attacks *Computers & Security* **24** (1), 31–43 2005

[40] Paulauskas, N & Garsva, E Computer System Attack Classification *Electronics and Electrical Engineering, Technology, Kaunas* **2** (66), 84–87, 2006

[41] Wu, Z *et al* A Taxonomy of Network and Computer Attacks Based on Responses In *Proc of the International Conference on Information Technology, Computer Engineering and Management Sciences* 26–29, IEEE Computer Society, Nanjing, Jiangsu 2011

[42] Kayacık, H G *et al* Selecting Features for Intrusion Detection A Feature Relevance Analysis on KDD 99 Intrusion Detection Datasets In *Proc of the Third Annual Conference on Privacy, Security and Trust (PST-2005)*, 2005

[43] Ghorbani, A A *et al Network Intrusion Detection and Prevention Concepts and Techniques* Advances in Information Security Springer-verlag 2009

[44] Uhlig S *Implications of Traffic Characteristics on Interdomain Traffic Engineering* Ph D thesis, Universit catholique de Louvain, 2004

[45] Ning, P & Jajodia S *Intrusion Detection Techniques*, H Bidgoli (Ed), The Internet Encyclopedia 2003

[46] Wikimedia F Intrusion Detection System http //en wikipedia org/wiki/Intrusion-detection_system, 2009

[47] Sundaram, A An Introduction to Intrusion Detection *Crossroads* **2** (4), 3–7, 1996

[48] Park B C *et al* Towards Automated Application Signature Generation for Traffic Identification In *Proc of the IEEE/IFIP Network Operations and Management Symposium Pervasive Management for Ubiquitous Networks and Services* 160–167, 2008

[49] Jacobson, V *et al* tcpdump URL `ftp //ftp ee lbl gov/tcpdump tar gz`

[50] Dainotti, A & Pescape A Plab A Packet Capture and Analysis Architecture, 2004

[51] McCanne S & Jacobson V The BSD Packet Filter A New Architecture for User Level Packet Capture In *Proc of the Winter 1993 USENIX Conference* 259–269, USENIX Association, 1993

[52] KDDcup99 Knowledge Discovery in Databases DARPA Archive http //www kdd ics uci edu/databases/kddcup99/task html, 1999

[53] Ertoz, L *et al Next Generation Data Mining*, chap Chapter 3 MINDS - Minnesota Intrusion Detection System 1–21, CRC press, 2004

[54] Lee, W & Stolfo, S J Data Mining Approaches for Intrusion Detection In *Proc of the 7th conference on USENIX Security Symposium - Volume 7* 1–7, USENIX Association, Berkeley, CA USA, 1998

[55] Joshi, M V *et al* Mining Needle in a Haystack Classifying Rare Classes via Two-phase Rule Induction *SIGMOD Records* **30** (2), 91–102, 2001

[56] Theiler, J & Cai, D M Resampling Approach for Anomaly Detection in Multispectral Images In *Proc of SPIE* vol 5093 230–240, SPIE, 2003

[57] Zhu, X Semi-Supervised Learning Literature Survey Tech Rep 1530, Department of Computer Sciences University of Wisconsin-Madison 2005

[58] Fujimaki, R *et al* An Approach to Spacecraft Anomaly Detection Problem Using Kernel Feature Space In *Proc of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining* 401–410, ACM, USA, 2005

[59] Zhong C & Li, N Incremental Clustering Algorithm for Intrusion Detection Using Clonal Selection In *Proc of the IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application* 326–331, IEEE Computer Society, Washington DC, USA 2008

[60] Hsu C C & Huang Y P Incremental Clustering of Mixed Data Based on Distance Hierarchy *Journal of Expert Systems with Applications* **35** (3), 1177–1185 2008

[61] Burbeck, K & Nadjm-tehrani, S ADWICE - Anomaly Detection with Real-time Incremental Clustering In *Proc of the 7th International Conference on Information Security and Cryptology, Seoul, Korea*, Springer Verlag, 2004

[62] Ren, F *et al* Using Density-Based Incremental Clustering for Anomaly Detection In *Proc of the International Conference on Computer Science and Software Engineering* 986–989 IEEE Computer Society, Washington DC, USA 2008

[63] Zhang, C. *et al.* A Mixed Unsupervised Clustering-Based Intrusion Detection Model. In *Proc. of the 3rd International Conference on Genetic and Evolutionary Computing.* 426–428, IEEE CS, USA, 2009.

[64] Portnoy, L. *et al.* Intrusion Detection with Unlabeled Data Using Clustering. In *Proc. of the ACM CSS Workshop on on Data Mining Applied to Security.* 5–8, Philadelphia, PA, 2001.

[65] Zhang, J. & Zulkernine, M. Anomaly Based Network Intrusion Detection with Unsupervised Outlier Detection. In *Proc. of the IEEE International Conference on Communications*, vol. 5. 2388–2393, 2006.

[66] Shon, T. & Moon, J. A Hybrid Machine Learning Approach to Network Anomaly Detection. *Information Sciences* **177**, 3799–3821, 2007.

[67] Bahrololum, M. *et al.* Anomaly Intrsion Detection Design Using Hybrid of Unsupervised and supervised Neural Network. *International Journal of Computer Networks & Communications* **1** (2), 26–33, 2009.

[68] Lesot, M. J. & Rifqi, M. Anomaly-based Network Intrusion Detection : Techniques, Systems and Challenges. *International Journal of Knowledge Engineering and Soft Data Paradigms* **1** (1), 63–84, 2009.

[69] 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), 2007.

[70] Choi, S. *et al.* A Survey of Binary Similarity and Distance Measures. *Journal of Systemics, Cybernetics and Informatics* **8** (1), 43–48, 2010.

[71] Lesot, M. J. *et al.* Similarity Measures for Binary and Numerical Data: A Survey. *International Journal of Knowledge Engineering and Soft Data Paradigms* **1** (1), 63–84, 2009.

[72] Boriah, S. *et al.* Similarity Measures for Categorical Data: A Comparative Evaluation. In *Proc. of the 8th SIAM International Conference on Data Mining.* 243–254, 2008.

[73] Gan, G. *et al. Data Clustering Theory, Algorithms and Applications*, SIAM, 2007.

[74] Hsu, C. C. & Wang, S. H An Integrated Framework for Visualized and Exploratory Pattern Discovery in Mixed Data. *IEEE Transactions on Knowledge and Data Engineering* **18** (2), 161–173, 2005.

[75] Dash, M. & Liu, H. Feature Selection for Classification. *Intelligent Data Analysis* **1**, 131–156, 1997.

[76] Chen, Y. *et al.* Survey and Taxonomy of Feature Selection Algorithms in Intrusion Detection System. In *Proc. of the Second SKLOIS Conference on Information Security and Cryptology.* 153–167, Springer-Verlag, 2006.

[77] Li, Y. *et al* Building Lightweight Intrusion Detection System Using Wrapper-based Feature Selection Mechanisms. *Computers & Security* **28** (6), 466–475, 2009.

[78] Nguyen, H. T. *et al.* Towards a Generic Feature-Selection Measure for Intrusion Detection. In *Proc. of the 20th International Conference on Pattern Recognition.* 1529–1532, 2010.

[79] Sung, A. H. & Mukkamala, S. Identifying Important Features for Intrusion Detection Using Support Vector Machines and Neural Networks. In *Proc. of the Symposium on Applications and the Internet.* 209–217, IEEE CS, USA, 2003.

[80] Peng, H. *et al.* Feature Selection Based on Mutual Information : Criteria of Max-Dependency, Max-Relevance, and Min-Redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **27** (8), 1226–1238, 2005.

[81] Amiri, F. *et al.* Mutual Information-based Feature Selection for Intrusion Detection Systems. *Journal of Network and Computer Applications* **34** (4), 1184–1199, 2011.

[82] Dunn, J. C. Well Separated Clusters and Optimal Fuzzy Partitions. *Journal of Cybernetics* **4** (1), 95–104, 1974.

[83] Davies, D. L. & Bouldin, D. W. A Cluster Separation Measure. *IEEE Transaction on Pattern Analysis and Machine Intelligence* **1** (2), 224–227, 1979.

[84] Hubert, L. & Schultz, J. Quadratic Assignment as a General Data Analysis Strategy. *British Journal of Mathematical and Statistical Psychology* **29** (2), 190–241, 1976.

[85] Baker, F. B & Hubert, L. J. Measuring the Power of Hierarchical Cluster Analysis. *Journal of American Statistics Association* **70** (349), 31–38, 1975.

[86] Rohlf, F. J. Methods of Comparing Classifications *Annual Review of Ecology and Systematics* **5** (1), 101–113, 1974.

[87] 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.

[88] Goodman, L. & Kruskal, W. Measures of Associations for Cross-validations. *Journal of American Statistics Association* **49**, 732–764, 1954.

[89] Jaccard, P. The Distribution of Flora in the Alpine Zone. *New Phytologist* **11** (2), 37–50. 1912.

[90] Rand, W M Objective Criteria for the Evaluation of Clustering Methods *Journal of the American Statistical Association* **66** (336), 846–850, 1971.

[91] Bezdek. J. C. Numerical Taxonomy with Fuzzy Sets *Journal of Mathematical Biology* **1** (1), 57–71, 1974.

[92] Bezdek, J. C. Cluster Validity with Fuzzy Sets. *Journal of Cybernetics* **3** (3), 58–78, 1974.

[93] Xie, X. L. & Beni, G. A Validity Measure for Fuzzy Clustering. *IEEE Transactions on Pattern Analysis and machine Intelligence* **13** (4), 841–847, 1991.

[94] Axelsson, S The Base-rate Fallacy and the Difficulty of Intrusion Detection. *ACM Transactions on Information and System Security* **3** (3), 186–205, 2000.

[95] Lippmann, R. P *et al* Evaluating Intrusion Detection Systems· The 1998 DARPA Offline Intrusion Detection Evaluation In *DARPA Information Survivability Conference and Exposition*, vol 2. 12–26, 2000.

[96] Joshi, M. V. *et al* Predicting Rare Classes . Can Boosting Make any Weak Learner Strong? In *Proc of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* 297–306, ACM, USA, 2002

[97] Dokas, P *et al.* Data Mining for Network Intrusion Detection. In *Proc. of the NSF Workshop on Next Generation Data Mining,* 2002

[98] Wang, Y. *Statistical Techniques for Network Security : Modern Statistically-Based Intrusion Detection and Protection*, Information Science Reference, IGI Publishing, Hershey, PA ,2008

[99] Weiss, S M & Zhang, T. *The Handbook of Data Mining*, chap. Performance Alanysis and Evaluation, 426–439, Lawrence Erlbaum Assoc, 2003.

[100] Provost, F. J. & Fawcett, T. Robust Classification for Imprecise Environments *Machine Learning* **42** (3), 203–231, 2001

[101] Maxion, R. A & Roberts, R R. Proper Use of ROC Curves in Intrusion/Anomaly Detection. Tech. Rep. CS-TR-871, School of Computing Science, University of Newcastle upon Tyne, 2004.

[102] Sekar, R. *et al* A High-performance Network Intrusion Detection System. In *Proc. of the 6th ACM Conference on Computer and Communications Security* 8–17, ACM, USA, 1999.

[103] Ampah, N. K. *et al.* An Intrusion Detection Technique Based on Continuous Binary Communication Channels. *International Journal of Security and Networks* **6** (2/3), 174–180, 2011

[104] Edgeworth, F Y On Discordant Observations. *Philosophical Magazine* **23** (5), 364–375, 1887.

[105] Hodge, V. & Austin. J. A Survey of Outlier Detection Methodologies. *Artificial Intelligence Review* **22** (2), 85–126, 2004.

[106] Nguyen, T. & Armitage, G. A Survey of Techniques for Internet Traffic Classification using Machine Learning. *IEEE Communications Surveys & Tutorials* **10** (4), 56–76, 2008.

[107] Agyemang, M. *et al* A Comprehensive Survey of Numeric and Symbolic Outlier Mining Techniques *Intelligence Data Analysis* **10** (6), 521–538, 2006.

[108] Ma, J. & Perkings, S. Online Novelty Detection on Temporal Sequences. In *Proc of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* 613–618, ACM, 2003

[109] Snyder, D. *Online Intrusion Detection Using Sequences of System Calls.* Master's thesis, Department of Computer Science, Florida State University, 2001.

[110] Rousseeuw, P. J & Leroy, A. M. *Robust Regression and Outlier Detection,* John Wiley & Sons, 1987

[111] Barnett, V & Lewis, T. *Outliers in Statistical Data,* John Wiley & Sons, 1994.

[112] Hawkins, D. *Identification of Outliers,* Chapman and Hall, New York, 1980

[113] Beckman, R. J & Cook, R. D. Outliers. *Technometrics* **25** (2), 119–149, 1983.

[114] Bakar, Z. *et al.* A Comparative Study for Outlier Detection Techniques in Data Mining. In *Proc of the IEEE Conference on Cybernetics and Intelligent Systems* 1–6, 2006

[115] Gogoi, P. *et al* A Survey of Outlier Detection Methods in Network Anomaly Identification. *The Computer Journal* **54** (5), 570–588, 2011

[116] Garcia-Teodoro, P. *et al.* Anomaly-based Network Intrusion Detection · Techniques, Systems and Challenges. *Computers & Security* **28** (1-2), 18–28, 2009.

[117] Callado, A. *et al.* A Survey on Internet Traffic Identification. *IEEE Communications Surveys & Tutorials* **11** (3), 37–52, 2009.

[118] Zhang, W. *et al.* A Survey of Anomaly Detection Methods in Networks. In *Proc. of the International Symposium on Computer Network and Multimedia Technology.* 1–3, 2009

[119] Sperotto, A. *et al.* An Overview of IP Flow-Based Intrusion Detection. *IEEE Communications Surveys & Tutorials* **12** (3), 343–356, 2010.

[120] Peng, T. *et al.* Survey of Network-based Defense Mechanisms Countering the DoS and DDoS Problems. *ACM Computing Surveys* **39** (1), 1–42, 2007.

[121] Al-Kuwaiti, M. *et al.* A Comparative Analysis of Network Dependability, Fault-tolerance, Reliability, Security, and Survivability. *IEEE Communications Surveys & Tutorials* **11** (2), 106–124, 2009.

[122] Donnet, B. *et al.* A Survey on Network Coordinates Systems, Design, and Security. *IEEE Communication Surveys & Tutorials* **12** (4), 488–503, 2010.

[123] Tavallaee, M. *et al.* Toward Credible Evaluation of Anomaly-based Intrusion Detection Methods. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews* **40** (5), 516–524, 2010.

[124] Daniel, B. *et al* ADAM: A Testbed for Exploring the Use of Data Mining in Intrusion Detection. *ACM SIGMOD Record* **30** (4), 15–24, 2001.

[125] Zhang, Z. *et al.* HIDE: a Hierarchical Network Intrusion Detection System Using Statistical Preprocessing and Neural Network Classification. In *Proc. of IEEE Man Systems and Cybernetics Information Assurance Workshop*, 2001.

[126] Anscombe, F. J. & Guttman, I. Rejection of Outliers. *Technometrics* **2** (2), 123–147, 1960

[127] Eskin, E. Anomaly Detection Over Noisy Data Using Learned Probability Distributions. In *Proc. of the 7th International Conference on Machine Learning.* 255–262, Morgan Kaufmann Publishers Inc., 2000.

[128] Desforges, M. *et al.* Applications of Probability Density Estimation to the Detection of Abnormal Conditions in Engineering. In *Proc. of Institute of Mechanical Engineers*, vol. 212. 687–703, 1998

[129] Manikopoulos, C. & Papavassiliou, S. Network Intrusion and Fault Detection: A Statistical Anomaly Approach. *IEEE Communications Magazine* **40** (10), 76–82, 2002.

[130] Chan, P K *et al* A Machine Learning Approach to Anomaly Detection Tech Rep CS-2003-06, Department of Computer Science Florida Institute of Technology, 2003

[131] Mahoney, M V & Chan, P K Learning Rules for Anomaly Detection of Hostile Network Traffic In *Proc of the 3rd IEEE International Conference on Data Mining*, IEEE CS, Washington, 2003

[132] Wang, K & Stolfo, S J Anomalous Payload-Based Network Intrusion Detection In *Proc of the Recent Advances in Intrusion Detection* 203–222, springer 2004

[133] Song, X *et al* Conditional Anomaly Detection *IEEE Trans on Knowledge and Data Engineering* **19** (5), 631–645 2007

[134] Chhabra P *et al* Distributed Spatial Anomaly Detection In *Proc of the 27th IEEE International Conference on Computer Communications* 1705–1713 2008

[135] Lu W & Ghorbani, A A Network Anomaly Detection Based on Wavelet Analysis *EURASIP Journal on Advances in Signal Processing* **2009** (837601) 2009

[136] Wattenberg, F S *et al* Anomaly Detection in Network Traffic Based on Statistical Inference and α-Stable Modeling *IEEE Transactions on Dependable and Secure Computing* **8** (4) 494–509, 2011

[137] Yu, M A Nonparametric Adaptive CUSUM Method and Its Application in Network Anomaly Detection *International Journal of Advancements in Computing Technology* **4** (1) 280–288, 2012

[138] Friedman N *et al* Bayesian Network Classifiers *Machine Learning* **29** (2-3), 131–163, 1997

[139] Kruegel, C *et al* Bayesian Event Classification for Intrusion Detection In *Proc of the 19th Annual Computer Security Applications Conference*, 2003

[140] Agrawal, R & Srikant R Fast Algorithms for Mining Association Rules in Large Databases In *Proc of the 20th International Conference on Very Large Data Bases* 487–499, Morgan Kaufmann San Francisco, CA USA, 1994

[141] Subramoniam, N *et al* Development of a Comprehensive Intrusion Detection System - Challenges and Approaches In *Proc of the 1st International Conference on Information Systems Security* 332–335, Kolkata, India, 2005

[142] Song, S *et al* Flow-based Statistical Aggregation Schemes for Network Anomaly Detection In *Proc of the IEEE International Conference on Networking, Sensing*, 2006

[143] Tong H *et al* Anomaly Internet Network Traffic Detection by Kernel Principle Component Classifier In *Proc of the 2nd International Symposium on Neural Networks*, vol LNCS 3498 476–481, 2005

[144] Gaddam, S R *et al* K-Means+ID3 A Novel Method for Supervised Anomaly Detection by Cascading K-Means Clustering and ID3 Decision Tree Learning Methods *IEEE Transactions on Knowledge and Data Engineering* **19** (3), 345–354, 2007

[145] Das, K *et al* Anomaly Pattern Detection in Categorical Datasets In *Proc of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* 169–176, ACM USA, 2008

[146] Lu W & Tong H Detecting Network Anomalies Using CUSUM and EM Clustering In *Proc of the 4th International Symposium on Advances in Computation and Intelligence* 297–308, Springer, 2009

[147] Qadeer, M A *et al* Network Traffic Analysis and Intrusion Detection Using Packet Sniffer In *Proc of the 2nd International Conference on Communication Software and Networks* 313–317 IEEE Computer Society, Washington, DC, USA, 2010

[148] Kang, I *et al* A Differentiated One-class Classification Method with Applications to Intrusion Detection *Expert Systems with Applications* **39** (4), 3899–3905, 2012

[149] Tsai, C. F. *et al.* Intrusion Detection by Machine Learning: A Review. *Expert Systems with Applications* **36** (10), 11994–12000, 2009.

[150] Abbes, T. *et al.* Efficient Decision Tree for Protocol Analysis in Intrusion Detection. *International Journal of Security and Networks* **5** (4), 220–235, 2010.

[151] Wagner, C. *et al.* Machine Learning Approach for IP-Flow Record Anomaly Detection. In *Proc. of the 10th International IFIP TC 6 conference on Networking - Volume Part I* 28–39, 2011.

[152] SchAolkopf, B. *et al* Estimating the Support of a High-dimensional Distribution. *Neural Computing* **13** (7), 1443–1471, 2001

[153] Su, M. Y. *et al.* A Real-time Network Intrusion Detection System for Large-scale Attacks Based on an Incremental Mining Approach. *Computers & Security* **28** (5), 301–309, 2009.

[154] Khan, L. *et al* A New Intrusion Detection System Using Support Vector Machines and Hierarchical Clustering. *The VLDB Journal* **16** (4), 507–521, 2007.

[155] Muda, Z. *et al.* A K-means and Naive-bayes Learning Approach for Better Intrusion Detection. *Information Technology Journal* **10** (3), 648–655. 2011

[156] Quinlan, J. R Induction of Decision Trees *Machine Learning* **1** (1), 81–106, 1986.

[157] Yu, H. & Kim, S *Handbook of Natural Computing*, chap SVM Tutorial - Classification, Regression and Ranking, Springer, 2003.

[158] Kuang, L. V. *DNIDS: A Dependable Network Intrusion Detection System Using the CSI-KNN Algorithm* Master's thesis, Queen's University Kingston, Ontario, Canada, 2007

[159] Bhuyan, M. H. *et al.* RODD: An Effective Reference-Based Outlier Detection Technique for Large Datasets. In *Advanced Computing*, vol. 133, 76–84, Springer, 2011.

[160] Lee, W. *et al.* Adaptive Intrusion Detection : A Data Mining Approach. *Artificial Intelligence Review* **14** (6), 533–567, 2000.

[161] Roesch, M. Snort-lightweight Intrusion Detection for Networks. In *Proc. of the 13th Systems Administration Conference*. 229–238, Usenix Association, Seattle, WA, USA, 1999.

[162] Neumann, B. Knowledge Management and Assistance Systems. http://kogs-www informatik.uni-hamburg.de/ neumann/, 2007.

[163] Zhang, Y. F *et al.* Distributed Intrusion Detection Based on Clustering. In *Proc. of the International Conference on Machine Learning and Cybernetics*, vol. 4. 2379–2383, 2005.

[164] Leung, K. & Leckie, C. Unsupervised Anomaly Detection in Network Intrusion Detection Using Clusters. In *Proc. of the 28th Australasian conference on Computer Science - Volume 38*. 333–342, Australian Computer Society, Darlinghurst, Australia, Australia, 2005.

[165] Sequeira, K & Zaki, M. ADMIT: Anomaly-based Data Mining for Intrusions. In *Proc. of the 8th ACM SIGKDD international conference on Knowledge discovery and data mining*. 386–395, ACM, New York, NY, USA, 2002

[166] Eskin, E. *et al. Applications of Data Mining in Computer Security*, chap. A Geometric Framework for Unsupervised Anomaly Detection: Detecting Intrusions in Unlabeled Data, Kluwer Academic, 2002.

[167] Zhuang, Z *et al.* Enhancing Intrusion Detection System with Proximity Information. *International Journal of Security and Networks* **5** (4), 207–219, 2010.

[168] 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.

[169] Chen, Z. & Chen, C. A Closed-Form Expression for Static Worm-Scanning Strategies. In *Proc. of the IEEE International Conference on Communications*. 1573–1577, IEEE CS, Beijing, China, 2008.

[170] Balajinath, B. & Raghavan, S. V. Intrusion Detection Through Learning Behavior Model. *Computer Communications* **24** (12), 1202–1212, 2001

[171] Khan, M. S. A. Rule based Network Intrusion Detection using Genetic Algorithm. *International Journal of Computer Applications* **18** (8), 26–29, 2011.

[172] Haykin, S. *Neural Networks*, Prentice Hall, New Jersey, 1999.

[173] 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.

[174] Carpenter, G. & Grossberg, S. Adaptive Resonance Theory. *CAS/CNS Technical Report Series* (008), 2010.

[175] Kohonen, T. The Self-organizing Map. *Proc. of the IEEE* **78** (9), 1464–1480, 1990.

[176] Cannady, J. Applying CMAC-Based On-Line Learning to Intrusion Detection. In *Proc. of the IEEE-INNS-ENNS International Joint Conference on Neural Networks*, vol. 5. 405–410, 2000.

[177] Lee, W & Xiang, D. Information-Theoretic Measures for Anomaly Detection. In *Proc. of the IEEE Symposium on Security and Privacy*. 130–143, IEEE Computer Society, Washington, DC, USA, 2001.

[178] Liu, G. *et al* A Hierarchical Intrusion Detection Model Based on the PCA Neural Networks. *Neurocomputing* **70** (7-9), 1561–1568, 2007.

[179] Sun, J. *et al* Intrusion Detection Method Based on Wavelet Neural Network. In *Proc. of the 2nd International Workshop on Knowledge Discovery and Data Mining*. 851–854, IEEE CS, USA, 2009.

[180] Labib, K. & Vemuri, R. NSOM: A Tool to Detect Denial of Service Attacks Using Self-Organizing Maps. Tech. Rep., Department of Applied Science University of California, Davis Davis, California, U.S.A., 2002.

[181] Bolzoni, D. *et al.* POSEIDON: a 2-tier Anomaly-based Network Intrusion Detection System. In *Proc. of the 4th IEEE International Workshop on Information Assurance*. 144–156, 2006.

[182] Mahoney, M. V. & Chan, P. K. PHAD: Packet Header Anomaly Detection for Indentifying Hostile Network Traffic. Tech Rep. CS-2001-04, Department of Computer Sciences, Florida Tech, Melbourne, FL 32901, 2001 URL http://cs.fit.edu/ tr/

[183] Dickerson, J. E. Fuzzy Network Profiling for Intrusion Detection. In *Proc. of the 19th International Conference of the North American Fuzzy Information Processing Society* 301–306, Atlanta, 2000

[184] Geramiraz, F. *et al* Adaptive Anomaly-Based Intrusion Detection System Using Fuzzy Controller *International Journal of Network Security* **14** (6), 352–361, 2012.

[185] Tajbakhsh, A. *et al.* Intrusion Detection Using Fuzzy Association Rules. *Applied Soft Computing* **9** (2), 462–469, 2009.

[186] Mabu, S. *et al* An Intrusion-Detection Model Based on Fuzzy Class-Association-Rule Mining Using Genetic Network Programming *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews* **41** (1), 130–139, 2011.

[187] Xian, J. Q. *et al* A Novel Intrusion Detection Method Based on Clonal Selection Clustering Algorithm In *Proc of the International Conference on Machine Learning and Cybernetics*, vol 6, IEEE Press, USA, 2005.

[188] Mohajerani, M *et al* NFIDS A Neuro-Fuzzy Intrusion Detection System. In *Proc. of the 10th IEEE International Conference on Electronics, Circuits and Systems*, vol. 1. 348–351, 2003.

[189] Pawlak, Z. Rough Sets. *International Journal of Parallel Programming* **11** (5), 341–356, 1982.

[190] Cai, Z. *et al.* A Rough Set Theory Based Method for Anomaly Intrusion Detection in Computer Network Systems. *Expert Systems* **20** (5), 251–259, 2003.

[191] Chimphlee, W. *et al.* Anomaly-Based Intrusion Detection using Fuzzy Rough Clustering. In *Proc. of the International Conference on Hybrid Information*

Technology, vol. 01. 329–334, IEEE Computer Society, Washington, DC, USA, 2006.

[192] Chen, R C *et al.* Using Rough Set and Support Vector Machine for Network Intrusion Detection System. In *Proc. of the 1st Asian Conference on Intelligent Information and Database Systems*. 465–470, IEEE Computer Society, Washington, DC, USA, 2009.

[193] Adetunmbi, A O. *et al* Network Intrusion Detection Based on Rough Set and k-Nearest Neighbour. *International Journal of Computing and ICT Research* **2** (1), 60–66, 2008.

[194] Visconti, A. & Tahayori, H. Artificial Immune System Based on Interval Type-2 Fuzzy Set Paradigm *Applied Soft Computing* **11** (6), 4055–4063, 2011.

[195] Noel, S. *et al.* Modern Intrusion Detection, Data Mining, and Degrees of Attack Guilt. In *Proc. of the International Conference on Applications of Data Mining in Computer Security*, Springer, 2002.

[196] Sekar, R. *et al.* Specification-based Anomaly Detection: A New Approach for Detecting Network Intrusions. In *Proc. of the 9th ACM Conference on Computer and Communications Security*. 265–274, 2002.

[197] Xu, X Sequential Anomaly Detection Based on Temporal Difference Learning: Principles, Models and Case Studies. *Applied Soft Computing* **10** (3), 859–867, 2010.

[198] Prayote, A *Knowledge Based Anomaly Detection.* Ph.D. thesis, School of Computer Science and Egineering, The University of New South Wales, 2007.

[199] Ilgun, K. *et al.* State Transition Analysis: A Rule-based Intrusion Detection Approach. *IEEE Transactions on Software Engineering* **21** (3), 181–199, 1995.

[200] Denning, D. E. & Neumann, P. G. Requirements and Model for IDES A Real-time Intrusion Detection System. Tech. Rep. 83F83-01-00, Computer Science Laboratory, SRI International, USA, 1985

[201] Anderson, D. *et al.* Detecting Unusual Program Behaviour Using the Statistical Component of the Next-generation Intrusion Detection Expert System (NIDES). Tech. Rep. SRIO-CSL-95-06, Computer Science Laboratory, SRI International, USA, 1995.

[202] Duffield, N. G. *et al.* Rule-Based Anomaly Detection on IP Flows. In *Proc. of the 28th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies.* 424–432, IEEE press, Rio de Janeiro, Brazil, 2009.

[203] Schapire, R. E. A Brief Introduction to Boosting In *Proc. of the 16th International Joint Conference on Artificial Intelligence.* 1401–1406, Morgan Kaufmann, 1999.

[204] Prayote, A. & Compton, P. Detecting Anomalies and Intruders. *AI 2006: Advances in Artificial Intelligence* 1084–1088, 2006.

[205] Edwards, G. *et al.* Prudent Expert Systems with Credentials: Managing the Expertise of Decision Support Systems. *International journal of biomedical computing* **40** (2), 125–132, 1995.

[206] Scheirer, W. & Chuah, M. C. Syntax vs. Semantics : Competing Approaches to Dynamic Network Intrusion Detection. *International Journal Security and Networks* **3** (1), 24–35, 2008.

[207] Estevez-Tapiador, J. M. *et al.* Stochastic Protocol Modeling for Anomaly based Network Intrusion Detection. In *Proc. of the 1st International Workshop on Information Assurance.* 3–12, IEEE CS, 2003.

[208] Hung, S. S. & Liu, D. S. M. A User-oriented Ontology-based Approach for Network Intrusion Detection. *Computer Standards & Interfaces* **30** (1-2), 78–88, 2008.

[209] Shabtai, A. *et al.* Intrusion Detection for Mobile Devices Using the Knowledge-based, Temporal Abstraction Method. *Journal of System Software* **83** (8), 1524–1537, 2010.

[210] Polikar, R Ensemble Based Systems in Decision Making. *IEEE Circuits System Magazine* **6** (3), 21–45, 2006.

[211] Borji, A. Combining Heterogeneous Classifiers for Network Intrusion Detection. In *Proc. of the 12th Asian Computing Science Conference on Advances in Computer Science: Computer and Network Security.* 254–260, Springer, 2007

[212] Giacinto, G. *et al* Intrusion Detection in Computer Networks by a Modular Ensemble of One-class Classifiers. *Information Fusion* **9** (1), 69–82, 2008.

[213] Rokach, L Ensemble-based Classifiers. *Artificial Intelligence Review* **33** (1-2), 1–39, 2010

[214] Noto, K *et al* Anomaly Detection Using an Ensemble of Feature Models. In *Proc of the IEEE International Conference on Data Mining.* 953–958, IEEE CS, USA, 2010.

[215] Mafra, P M. *et al* Octopus-IIDS: An Anomaly Based Intelligent Intrusion Detection System In *Proc. of the IEEE Symposium on Computers and Communications.* 405–410, IEEE CS, USA, 2010.

[216] Chebrolu, S. *et al* Feature Reduction and Ensemble Design of Intrusion Detection Systems. *Computers & Security* **24** (4), 295–307, 2005

[217] Breiman, L. *et al* *Classification and Regression Trees*, Wadsworth and Brooks, Monterey, CA, 1984

[218] Perdisci, R. *et al.* Using an Ensemble of One-Class SVM Classifiers to Harden Payload-based Anomaly Detection Systems. In *Proc. of the 6th International Conference on Data Mining.* 488–498, IEEE CS, USA, 2006.

[219] Folino, G. *et al* An Ensemble-based Evolutionary Framework for Coping with Distributed Intrusion Detection. *Genetic Programming and Evolvable Machines* **11** (2), 131–146, 2010.

[220] Nguyen, H. H. *et al.* An Efficient Local Region and Clustering-based Ensemble System for Intrusion Detection. In *Proc of the 15th Symposium on International Database Engineering & Applications* 185–191, ACM, USA, 2011.

[221] Rehak, M. *et al.* CAMNEP: Agent-based Network Intrusion Detection System. In *Proc. of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems: Industrial Track.* 133–136, IFAAMS, Richland, SC, 2008.

[222] Perdisci, R. *et al.* McPAD. A Multiple Classifier System for Accurate Payload-based Anomaly Detection. *Computer Networks* **53** (6), 864–881, 2009.

[223] Giacinto, G. *et al.* Fusion of Multiple Classifiers for Intrusion Detection in Computer Networks. *Pattern Recognition Letters* **24** (12), 1795–1803, 2003.

[224] Shifflet, J. A Technique Independent Fusion Model For Network Intrusion Detection. In *Proc. of the Midstates Conference on Undergraduate Research in Computer Science and Mathematics*, vol. 3. 13–19, 2005.

[225] Parikh, D. & Chen, T. Data Fusion and Cost Minimization for Intrusion Detection. *IEEE Transactions on Information Forensics and Security* **3** (3), 381–389, 2008.

[226] Zhi-dong, L. *et al.* Decision-level Fusion Model of Multi-source Intrusion Detection Alerts. *Journal on Communications* **32** (5), 121–128, 2011.

[227] Yan, R. & Shao, C. Hierarchical Method for Anomaly Detection and Attack Identification in High-speed Network. *Information Technology Journal* **11** (9), 1243–1250, 2012.

[228] Chatzigiannakis, V. *et al.* Data Fusion Algorithms for Network Anomaly Detection: Classification and Evaluation In *Proc of the 3rd International Conference on Networking and Services.* 50–57, IEEE CS, Greece, 2007.

[229] Gong, W. *et al.* A Neural Network Based Intrusion Detection Data Fusion Model. In *Proc. of the 3rd International Joint Conference on Computational Science and Optimization - Volume 02.* 410–414, IEEE CS, USA, 2010.

[230] Ariu, D. *et al.* HMMPayl: An Intrusion Detection System Based on Hidden Markov Models. *Computers & Security* **30** (4), 221–241, 2011.

[231] Arumugam, M. *et al.* Implementation of Two Class Classifiers for Hybrid Intrusion Detection. In *Proc. of the International Conference on Communication and Computational Intelligence.* 486–490, 2010.

[232] Herrero, A. *et al.* RT-MOVICAB-IDS: Addressing Real-time Intrusion Detection. *Future Generation Computer Systems* **29** (1), 250–261, 2011.

[233] Locasto, M. E. *et al.* FLIPS: Hybrid Adaptive Intrusion Prevention. In *Recent Advances in Intrusion Detection.* 82–101, 2005.

[234] Peddabachigari, S. *et al.* Modeling Intrusion Detection System Using Hybrid Intelligent Systems. *Journal of Network and Computer Applications* **30** (1), 114–132, 2007.

[235] Zhang, J. *et al.* Random-Forests-Based Network Intrusion Detection Systems. *IEEE Transactions on Systems, Man, and Cybernetics: Part C* **38** (5), 649–659, 2008.

[236] Tong, X. *et al.* A Research Using Hybrid RBF/Elman Neural Networks for Intrusion Detection System Secure Model. *Computer Physics Communications* **180** (10), 1795–1801, 2009.

[237] Yu, X. A New Model of Intelligent Hybrid Network Intrusion Detection System. In *Proc. of the International Conference on Bioinformatics and Biomedical Technology.* 386–389, IEEE CS, 2010.

[238] Selim, S. *et al.* Hybrid Multi-level Intrusion Detection System. *International Journal of Computer Science and Information Security* **9** (5), 23–29, 2011.

[239] Soule, A. *et al.* Combining Filtering and Statistical Methods for Anomaly Detection. In *Proc. of the 5th ACM SIGCOMM conference on Internet Measurement.* 31–31, USENIX Association, Berkeley, CA, USA, 2005.

[240] Sperotto, A. *et al.* A Labeled Data Set for Flow-Based Intrusion Detection. In *Proc of the 9th IEEE International Workshop on IP Operations and Management*, IPOM '09. 39–50, Springer-Verlag, Venice, Italy, 2009.

[241] Lazarevic, A. *et al.* A Comparative Study of Anomaly Detection Schemes in Network Intrusion Detection. In *Proc. of the 3rd SIAM International Conference on Data mining*, SIAM, 2003.

[242] Cemerlic, A. *et al.* Network Intrusion Detection Based on Bayesian Networks. In *Proc. of the 20th International Conference on Software Engineering and Knowledge Engineering*, SEKE'08. 791–794, KSI, San Francisco, CA, USA, 2008.

[243] Thomas, C. *et al.* Usefulness of DARPA Dataset for Intrusion Detection System Evaluation. In *Proc. of the Data Mining, Intrusion Detection, Information Assurance, and Data Networks Security, 6973*, SPIE, Orlando, FL, 2008.

[244] Denning, D. E. An Intrusion-Detection Model. *IEEE Transactions on Software Engineering* **13** (2), 222–232, 1987.

[245] Stolfo, S. J. *et al.* Cost-Based Modeling for Fraud and Intrusion Detection: Results from the JAM Project. In *Proc. of the DARPA Information Survivability Conference and Exposition*, vol. 2. 130–144, IEEE CS, USA, 2000.

[246] Kendall, K. *A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems.* Master's thesis, MIT, 1999.

[247] Delooze, L ' *Applying Soft-Computing Techniques to Intrusion Detection.* Ph.D. thesis, Computer Science Department, University of Colorado, Colorado Springs, 2005.

[248] Tavallaee, M. *et al.* A Detailed Analysis of the KDD CUP 99 Data Set. In *Proc. of the 2nd IEEE International Conference on Computational Intelligence for Security and Defense Applications.* 53–58, IEEE Press, USA, 2009.

[249] NSL-KDD. NSL-KDD Data Set for Network-based Intrusion Detection Systems. http://iscx.cs.unb.ca/NSL-KDD/, 2009.

[250] MIT Lincoln Lab, Information Systems Technology Group. DARPA Intrusion Detection Data Sets. http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/data/2000data.html, 2000.

[251] Defcon. The Shmoo Group. http://cctf.shmoo.com/, 2011.

[252] CAIDA. The Cooperative Analysis for Internet Data Analysis. http://www.caida.org, 2011.

[253] LBNL. Lawrence Berkeley National Laboratory and ICSI, LBNL/ICSI Enterprise Tracing Project. http://www.icir.org/enterprise-tracing/, 2005.

[254] Pang, R. *et al.* The Devil and Packet Trace Anonymization. *SIGCOMM Computer Communication Review* **36** (1), 29–38, 2006.

[255] *A First Look at Modern Enterprise Traffic*, USENIX Association, Berkeley, CA, USA, 2005.

[256] CACE Technologies. Winpcap. http://www.winpcap.org.

[257] symantec.com. Symantec Security Response. http://securityresponse.symantec.com/avcenter.

[258] UNIBS. University of Brescia Dataset. http://www.ing.unibs.it/ntw/tools/traces/, 2009.

[259] Shiravi, A. *et al.* Towards Developing a Systematic Approach to Generate Benchmark Datasets for Intrusion Detection. *Computers & Security* **31** (3), 357–374, 2012.

[260] Song, J. *et al.* Description of Kyoto University Benchmark Data. http://www.takakura.com/Kyoto_data/BenchmarkData-Description-v3.pdf, 2006.

[261] Mahoney, M. V. & Chan, P. K. An Analysis of the 1999 DARPA/Lincoln Laboratory Evaluation Data for Network Anomaly Detection. In *Proc. of the 6th International Symposium on Recent Advances in Intrusion Detection.* 220–237, Springer, 2003.

[262] McHugh, J. Testing Intrusion Detection Systems: A Critique of the 1998 and 1999 DARPA Intrusion Detection System Evaluations as Performed by Lincoln Laboratory. *ACM Transactions on Information and System Security* **3** (4), 262–294, 2000.

[263] Mell, P. *et al.* An Overview of Issues in Testing Intrusion Detection Systems. http://citeseer.ist.psu.edu/621355.html, 2003.

[264] Gogoi, P. *et al.* Packet and Flow-based Network Intrusion Dataset. In *Proc. of the 5th International Conference on Contemporary Computing*, vol. LNCS-CCIS 306. 322–334, Springer, 2012.

[265] Xu, J. & Shelton, C. R. Intrusion Detection using Continuous Time Bayesian Networks. *Journal of Artificial Intelligence Research* **39**, 745–774, 2010.

[266] Choo, K. K. R. The Cyber Threat Landscape: Challenges and Future Research Directions. *Computers & Security* **30** (8), 719–731, 2011.

[267] Mishra, B. K. & Ansari, G. M. Differential Epidemic Model of Virus and Worms in Computer Network. *International Journal of Network Security* **14** (3), 149–155, 2012.

[268] Braynov, S. & Jadliwala, M. Detecting Malicious Groups of Agents. In *Proc. of the 1st IEEE Symposium on Multi-Agent Security and Survivability*. 90–99, IEEE CS, 2004.

[269] Jung, J. *et al.* Fast Portscan Detection Using Sequential Hypothesis Testing. In *Proc. of the IEEE Symposium on Security and Privacy*. 211–225, IEEE Computer Society, Oakland, CA, 2004.

[270] De Vivo, M. *et al.* A Review of Port Scanning Techniques. *SIGCOMM Computer Communication Review* **29** (2), 41–48, 1999.

[271] Panjwani, S. *et al.* An Experimental Evaluation to Determine if Port Scans are Precursors to an Attack. In *Proc. of the International Conference on Dependable Systems and Networks*. 602–611, IEEE Computer Society, Washington, DC, USA, 2005.

[272] Mateti, P. Lecture Notes on Internet Security, 2010. Wright State University.

[273] hybrid@hotmail.com. Distributed Information Gathering. *Phrack Magazine, Article 9* **9** (55), 1999.

[274] Falletta, V. & Ricciato, F. Detecting Scanners: Empirical Assessment on 3G Network. *International Journal of Network Security* **9** (2), 143–155, 2009.

[275] Zhang, H. L. *Agent-based Open Connectivity for Decision Support Systems.* Ph.D. thesis, School of Computer Science and Mathematics, Victoria University, 2007.

[276] Ensafi, R. *et al.* Idle Port Scanning and Non-interference Analysis of Network Protocol Stacks Using Model Checking, 2010. Proc. of the 19th USENIX Security Symposium.

[277] Staniford-Chen, S. *et al.* GrIDS: A Graph Based Intrusion Detection System for Large Networks. In *Proc. of the 19th National Information Systems Security Conference.* 361–370, NIST, CSRC, Baltimore, MD, USA, 1996.

[278] Gregr, M. Portscan Detection Using NetFlow Data. In *Student EEICT 2010*, 602 00 Brno, Czech Republic, 2010.

[279] Staniford, S. *et al.* Practical Automated Detection of Stealthy Portscans. *Journal of Computer Security* **10** (1-2), 105–136, 2002.

[280] Yegneswaran, V. *et al.* Internet Intrusions: Global Characteristics and Prevalence. *SIGMETRICS Performance Evaluation Review* **31** (1), 138–147, 2003.

[281] Heberlein, T. *et al.* A Network Security Monitor. In *Proc. of the IEEE Symposium on Research in Security and Privacy.* 296–304, IEEE Computer Society, Oakland, California, USA, 1990.

[282] Fyodor. The Art of Port Scanning. *Phrack Magazine, Article 11* **7** (51), 1997.

[283] QoSient. Argus. http://www.qosient.com/argus/.

[284] Leckie, C. & Kotagiri, R. A Probabilistic Approach to Detecting Network Scans. In *Proc. of the IEEE Network Operations and Management Symposium.* 359–372, IEEE Computer Society, Florence, Italy, 2002.

[285] Kim, H. *et al.* Detecting Network Portscans Through Anomaly Detection. In *Proc. of SPIE on Detecting network portscans through anomaly detection*, vol. 5429. 254–263, SPIE, Orlando, FL, USA, 2004.

[286] Ertoz, L. *et al.* Detection of Novel Network Attacks Using Data Mining. In *Proc. of the ICDM Workshop on Data Mining for Computer Security.* 30–39, Florida Tech, Melbourne, Florida, USA, 2003.

[287] Gates, C. *et al.* Scan Detection on Very Large Networks Using Logistic Regression Modeling. In *Proc. of the 11th IEEE Symposium on Computers and Communications.* 402–408, IEEE Computer Society, Pula-Cagliari, Sardinia, Italy, 2006.

[288] Udhayan, J. *et al.* Reconnaissance Scan Detection Heuristics to Disrupt the Pre-attack Information Gathering. In *Proc. of the International Conference on Network and Service Security.* 1–5, IEEE Computer Society, ESIEA-9, 75005 Paris, 2009.

[289] Gyorgy, S. U. *et al.* Scan Detection: A Data Mining Approach. In *Proc. of the Sixth SIAM International Conference on Data Mining.* 118–129, SIAM, Sutton Place Hotel, Newport Beach, CA, 2005.

[290] Treurniet, J. A Network Activity Classification Schema and Its Application to Scan Detection. *IEEE/ACM Transactions on Networking* **19** (5), 1396–1404, 2011.

[291] Gates, C. *Co-ordinated Port Scans: A Model, A Detector and An Evaluation Methodology.* Ph.D. thesis, Dalhousie University, Halifax, Nova Scotia, 2006.

[292] Paxson, V. Bro: A System for Detecting Network Intruders in Real-time. In *Proc. of the the 7th USENIX Security Symposium.* 2435–2463, Usenix Association, San Antonio, Texas, 1998.

[293] *The OSU Flow-tools Package and CISCO NetFlow Logs*, USENIX Association, Berklay, CA, USA, 2000.

[294] Zhang, Y. & Fang, B. A Novel Approach to Scan Detection on the Backbone. In *Proc. of the Sixth International Conference on Information Technology:*

New Generations. 16–21, IEEE Computer Society, Washington, DC, USA, 2009.

[295] Sridharan, A. *et al* Connectionless Port Scan Detection on the Backbone In *Proc. of the 25th IEEE International Conference on Performance, Computing, and Communications*. 567–576, IEEE Computer Society, Phoenix, AZ, USA, 2006.

[296] Gadge, J. & Patil, A. A Port Scan Detection. In *Proc. of 16th IEEE International Conference on Networks*. 1–6, IEEE Computer Society, Habitat World, IHC, New Delhi, India, 2008

[297] Zadeh, L. A. Fuzzy Logic, Neural Networks, and Soft Computing. *Communication of the ACM* **37** (3), 77–84, 1994.

[298] Chen, J.-j. & Cheng, X -j A Novel Fast Port Scan Method Using Parthenogenetic Algorithm. In *Proc. of the 2nd IEEE International Conference on Computer Science and Information Technology*. 219–222, IEEE Computer Society, Los Alamitos, CA, USA, 2009

[299] Liu, D. *et al.* Network Traffic Analysis Using Refined Bayesian Reasoning to Detect Flooding and Port Scan Attacks. In *Proc of the International Conference on Advanced Computer Theory and Engineering*. 1000–1004, IEEE Computer Society, Phuket, Thailand, 2008

[300] Shafiq, M. Z. *et al.* A Comparative Study of Fuzzy Inference Systems, Neural Networks and Adaptive Neuro Fuzzy Inference Systems for Portscan Detection. In *Proc. of the conference on Applications of evolutionary computing*. 52–61, Springer-Verlag, 2008.

[301] Oke, G. & Loukas, G. A Denial of Service Detector Based on Maximum Likelihood Detection and the Random Neural Network. *The Computer Journal* **50** (6), 717–727, 2007.

[302] Kim, J. & Lee, J.-H. A Slow Port Scan Attack Detection Mechanism Based on Fuzzy Logic and a Stepwise Policy. In *Proc. of the IET 4th International Con-*

ference on Intelligent Environments. 1–5, IEEE Computer Society, University of Washington, Seattle, USA, 2008.

[303] Mueller, C. *et al.* Interactive Visualization for Network and Port Scan Detection In *LNCS Lecture Notes on Recent Advances in Intrusion Detection*, vol. 3858. 265–283, Springer-Verlag, Seattle, WA, USA, 2006

[304] McPherson, J *et al.* PortVis: A Tool for Port-based Detection of Security Events. In *Proc of CCS Workshop on Visualization and Data Mining for Computer Security.* 73–81, ACM, Washington, DC, USA, 2004

[305] Musa, S. & Parish, D. J Visualising Communication Network Security Attacks In *Proc of the 11th International Conference on Information Visualization* 726–733, IEEE Computer Society, Washington, DC, USA, 2007.

[306] Lee, A. J. *et al.* Searching for Open Windows and Unlocked Doors: Port Scanning in Large-scale Commodity Clusters. In *Proceedings of the IEEE International Symposium on Cluster Computing and the Grid,* vol 1. 146–151, IEEE Computer Society, Cardiff, UK, 2005

[307] Yurcik, W. *et al.* NVisionCC: A Visualization Framework for High Performance Cluster Security. In *Proc. of the ACM workshop on Visualization and data mining for computer security.* 133–137, ACM, New York, NY, USA, 2004

[308] Jiawan, Z *et al* A Novel Visualization Approach for Efficient Network Scans Detection. In *Proc. of the International Conference on Security Technology.* 23–26, IEEE Computer Society, Horizon Resort, Sanya, Hainan Island, China, 2008.

[309] Cheng, M. *et al.* Visualizing Graph Features for Fast Port Scan Detection. In *Proc of the 8th Annual Cyber Security and Information Intelligence Research Workshop.* 30:1–30:4, ACM, 2013.

[310] Porras, P. & Valdes, A. Live Traffic Analysis of TCP/IP Gateways. In *Proc. of the Internet Society Symposium on Network and Distributed System Security,* Internet Society, San Diego. 1998.

[311] Kato, N. *et al.* A Real-time Intrusion Detection System (IDS) for Large Scale Networks and Its Evaluations. *IEICE Transactions on Communications* **E82-B** (11), 1817–1825, 1999.

[312] Streilein, W. W. *et al.* Improved Detection of Low-profile Probe and Denial-of-Service Attacks. In *Proc. of the Workshop on Statistical and Machine Learning Techniques in Computer Intrusion Detection.* 11–13, Baltimore, Maryland, USA, 2002.

[313] Abdullah, K. *et al.* Visualizing Network Data for Intrusion Detection. In *Proc. of the Sixth Annual IEEE SMC Workshop on Information Assurance.* 100–108, IEEE Computer Society, West Point, NY, USA, 2005.

[314] Green, J. *et al.* Analysis Techniques for Detecting Coordinated Attacks and Probes. In *Proc. of the Workshop on Intrusion Detection and Network Monitoring.* 1–9, Usenix Association, Santa Clara, California, USA, 1999.

[315] Robertson, S. *et al.* Surveillance Detection in High Bandwidth Environments. In *Proc. of the DARPA DISCEX III Conference.* 130–139, IEEE Computer Society, Washington, DC, 2003.

[316] Whyte, D. *Network Scanning Detection Strategies for Enterprise Networks.* Ph.D. thesis, School of Computer Science, Carleton University, 2008.

[317] Singh, H. & Chun, R. Distributed Port Scan Detection. In *Handbook of Information and Communication Security*, 221–234, 2010, Springer-Verlag.

[318] Curtis, A. C. J. *et al.* A Methodology for Using Intelligent Agents to Provide Automated Intrusion Response. In *Proc. of the IEEE Workshop on Information Assurance and Security.* 110–116, IEEE Computer Society, United States Military Academy, West Point, NY, 2000.

[319] *Scan Attack Detection Based on Distributed Cooperative Model*, IEEE Computer Society, Xi'an, China, 2008.

[320] Conti, G. & Abdullah, K. Passive Visual Fingerprinting of Network Attack Tools. In *Proc. of CCS Workshop on Visualization and Data Mining for Computer Security.* 45–54, ACM, Washington, DC, USA, 2004.

[321] Stockinger, K. *et al* Detecting Distributed Scans Using High-performance Query-driven Visualization In *Proc of the 2006 ACM/IEEE conference on Supercomputing*, ACM, 2006

[322] Baldoni, R. *et al* *Distributed Computing and Networking*, vol. LNCS vol 7730, chap. Collaborative Detection of Coordinated Port Scans, 102–117, Springer Berlin Heidelberg, 2013.

[323] Zhang, Z. *et al.* An Observation-Centric Analysis on the Modeling of Anomaly-based Intrusion Detection *International Journal of Network Security* **4** (3), 292–305, 2007

[324] Liu, Z. *et al* A Method for Locating Digital Evidences with Outlier Detection Using Support Vector Machine *International Journal of Network Security* **6** (3), 301–308, 2008.

[325] Prakobphol, K & Zhan, J. A Novel Outlier Detection Scheme for Network Intrusion Detection Systems In *Proc of the International Conference on Information Security and Assurance.* 555–560, IEEE, Washington, DC, USA, 2008.

[326] Shyu, M. L. *et al* A Novel Anomaly Detection Scheme Based on Principal Component Classifier In *Proc of the IEEE Foundations and New Directions of Data Mining Workshop, in conjunction with ICDM'03.* 171–179, 2003

[327] Schneier, B. *Applied Cryptography : Protocols, Algorithms, and Source Code in C*, John Wiley & Sons, New York, USA, 1995, 2nd edn.

[328] Bezdek, J. C. *et al.* FCM: The Fuzzy C-means Clustering Algorithm. *Computers & Geosciences* **10** (2-3), 191 – 203, 1984.

[329] Knorr, E. M. & Ng, R. T. Algorithms for Mining Distance-Based Outliers in Large Datasets. In *Proc. of the 24th International Conference on Very Large Data Bases.* 392–403, Morgan Kaufmann, USA, 1998.

[330] Liu, H. *et al.* On-line Outlier Detection and Data Cleaning. *Computers and Chemical Engineering* **28** (9), 1635–1647, 2004

[331] Szeto, C.-C. & Hung, E. Mining Outliers with Faster Cutoff Update and Space Utilization. *Pattern Recognition Letters* **31** (11), 1292–1301, 2010.

[332] Xue, Z. *et al.* Semi-supervised Outlier Detection Based on Fuzzy Rough C-means Clustering. *Mathematics and Computers in Simulation* **80** (9), 1911–1921, 2010

[333] Tavallaee, M. *et al.* A Novel Covariance Matrix Based Approach for Detecting Network Anomalies. In *Proc. of the Communication Networks and Services Research Conference.* 75–81, IEEE, Washington, DC, USA, 2008.

[334] Lan, Z. *et al* Toward Automated Anomaly Identification in Large-Scale Systems *IEEE Transactions on Parallel and Distributed Systems* **21**, 174–187, 2010.

[335] Barbara, D. *et al* ADAM: Detecting Intrusions by Data Mining In *Proc. of the IEEE Workshop on Information Assurance and Security* 11–16, West Point, NY, 2001.

[336] Mukkamala, S. *et al* Intrusion Detection Using Neural Networks and Support Vector Machines. In *Proc. of the International Joint Conference on Neural Networks*, vol. 2. 1702–1707, IEEE, 2002.

[337] Wang, B. *et al* Outlier Detection over Sliding Windows for Probabilistic Data Streams. *Journal of Computer Science and Technology* **25** (3), 389–400, 2010.

[338] Barbara, D *et al.* Detecting Novel Network Intrusions Using Bayes Estimators. In *Proc. of the First SIAM Conference on Data Mining*, Chicago, IL, 2001.

[339] Knorr, E. M *et al.* Distance-based Outliers: Algorithms and Applications. *VLDB Journal* **8** (3-4), 237–253, 2000.

[340] Angiulli, F *et al.* Distance-Based Detection and Prediction of Outliers. *IEEE Transactions on Knowledge and Data Engineering* **18** (2), 145–160, 2006.

[341] Jiang, F. *et al* A Hybrid Approach to Outlier Detection Based on Boundary Region *Pattern Recognition Letters* **32** (14), 1860–1870, 2011.

[342] 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, 2010.

[343] Zhang, Y. *et al.* LDBOD: A Novel Local Distribution Based Outlier Detector. *Pattern Recognition Letters* **29** (7), 967–976, 2008.

[344] Shaari, F. *et al.* Outlier Detection Based on Rough Sets Theory. *Intelligent Data Analysis* **13** (2), 191–206, 2009.

[345] Bhuyan, M. H. *et al.* NADO : Network Anomaly Detection Using Outlier Approach. In *Proc. of the International Conference on Communication, Computing & Security*. 531–536, ACM, Odisha, India, 2011.

[346] Agrawal, A. Local Subspace Based Outlier Detection. In *Communication in Computer and Information Science*, vol. 40. 149–157, Springer, 2009.

[347] Orair, G. H. *et al.* Distance-based Outlier Detection: Consolidation and Renewed Bearing. *Proc. of the VLDB Endowment* **3** (1-2), 1469–1480, 2010.

[348] Kohavi, R. & John, G. H. Wrappers for Feature Subset Selection. *Artificial Intelligence - Special issue on relevance* **97** (1-2), 273–324, 1997.

[349] Olusola, A. A. *et al.* Analysis of KDD99 Intrusion Detection Dataset for Selection of Relevance Features. In *Proc. of the World Congress on Engineering and Computer Science*, vol. I. 162–168, IAENG, San Francisco, USA, 2010.

[350] Hartigan, J. A. & Wong, M. A. Algorithm AS 136: A K-means Clustering Algorithm. *Journal of the Royal Statistical Society, Series C (Applied Statistics)* **28** (1), 100–108, 1979.

[351] Ester, M. *et al.* A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proc. of Second International Conference on Knowledge Discovery and Data Mining*. 226–231, AAAI Press, Portland, Oregon, 1996.

[352] Andersen, R. *et al.* Overlapping Clusters for Distributed Computation. In *Proc. of the fifth ACM international conference on Web search and data mining.* 273–282, ACM, New York, NY, USA, 2012.

[353] Jain, A. K. & Dubes, R. C. *Algorithms for Clustering Data*, Prentice-Hall, Upper Saddle River, NJ, USA, 1988.

[354] Bezdek, J. C. *et al.* FCM: The Fuzzy C-means Clustering Algorithm. *Computers & Geosciences* **10** (2-3), 191–203, 1984.

[355] Gogoi, P. *et al.* Packet and Flow Based Network Intrusion Datasets. In *Proc. of the 5th International Conference on Contemporary Computing*, vol. LNCS-CCIS 306. 322–334, Springer-Verlag, Noida, India, 2012.

[356] Frank, A. & Asuncion, A. UCI Machine Learning Repository, 2010. URL `http://archive.ics.uci.edu/ml`.

[357] Moonesinghe, H. D. K. & Tan, P. N. OutRank: A Graph-Based Outlier Detection Framework Using Random Walk. *International Journal on Artificial Intelligence Tools* **17** (1), 19–36, 2008.

[358] Bhuyan, M. H. *et al.* An Effective Unsupervised Network Anomaly Detection Method. In *Proc. of the International Conference on Advances in Computing, Communications and Informatics.* 533–539, ACM, New York, NY, USA, 2012.

[359] SNORT. Open Source Network Intrusion Prevention and Detection System. http://www.snort.org/, 2010.

[360] BRO. Unix-based Network Intrusion Detection System. http://bro-ids.org/, 2011.

[361] Lange, T. *et al.* Stability-based Validation of Clustering Solutions. *Neural Comput.* **16** (6), 1299–1323, 2004.

[362] Mufti, B. G. *et al.* Determining the Number of Groups from Measures of Cluster Stability. In *Proc. of the Applied Stochastic Models and Data Analysis.* 404–413, 2005.

[363] Ben-David, S. *et al.* A Sober Look at Clustering Stability. In *COLT*. 5–19, 2006.

[364] Das, A. K. & Sil, J. Cluster Validation Method for Stable Cluster Formation. *Canadian Journal on Artificial Intelligence, Machine Learning and Pattern Recognition* **1** (3), 26–41, 2010.

[365] Halkidi, M. *et al.* On Clustering Validation Techniques. *Journal of Intelligent Information Systems* **17** (2-3), 107–145, 2001.

[366] Brock, G. *et al.* clValid: An R Package for Cluster Validation. *Journal of Statistical Software* **25** (4), 1–22, 2008.

[367] Jun, S. An Ensemble Method for Validation of Cluster Analysis. *International Journal of Computer Science Issues* **8** (6), 26–30, 2011.

[368] Bhuyan, M. H. *et al.* NADO: Network Anomaly Detection Using Outlier Approach. In *Proc. of the ACM International Conference on Communication, Computing & Security*. 531–536, ACM, USA, 2011.

[369] Renyi, A. On Measures of Entropy and Information. In *Berkeley Symposium Mathematics, Statistics, and Probability*. 547–561, 1960.

[370] Principe, J. C. *et al.* *Information Theoretic Learning*, chap. Unsupervised Adaptive Filtering, 265–319, John Wiley and Sons, New York, 2000.

[371] Jenssen, R. *et al.* Clustering Using Renyi's Entropy. In *Proc. of the joint international conference on neural networks*. 523–528, 2003.

[372] Quinlan, J. R. *C4.5: Programs for Machine Learning*, Morgan Kaufmann, San Francisco, CA, USA, 1993.

[373] Clark, P. & Niblett, T. The CN2 Induction Algorithm. *Machine Learning* **3** (4), 261–283, 1989.

[374] Daniel, W. W. *Biostatistics: A Foundation for Analysis in the Health Sciences*, John Wiley & Sons, New York, 1987.

[375] Lin, S. & Chiueh, T. C. , A Survey on Solutions to Distributed Denial of Service Attacks. Tech. Rep. TR201, Department of Computer Science, State University of New York, Stony Brook, http://www.ecsl.cs.sunysb.edu/tr/TR201.pdf, 2006.

[376] Specht, S. M. & Lee, R. B. Distributed Denial of Service: Taxonomies of Attacks, Tools, and Countermeasures. In *Proc. of the ISCA 17th International Conference on Parallel and Distributed Computing Systems*. 543–550, San Francisco, California, USA, 2004.

[377] Batishchev, A M. LOIC(Low Orbit Ion Cannon). http://sourceforge.net/projects/loic/, 2004.

[378] Beitollahi, H. & Deconinck, G. Analyzing Well-known Countermeasures Against Distributed Denial of Service Attacks. *Computer Communication* **35** (11), 1312–1332, 2012

[379] Xiang, Y. *et al.* Low-Rate DDoS Attacks Detection and Traceback by Using New Information Metrics. *IEEE Trans. on Information Forensics and Security* **6** (2), 426–437, 2011.

[380] Yu, S. *et al* Information Theory Based Detection Against Network Behavior Mimicking DDoS Attacks. *IEEE Communications Letters* **12** (4), 319–321, 2008.

[381] Wang, H. *et al.* Defense Against Spoofed IP Traffic Using Hop-count Filtering. *IEEE/ACM Trans. on Networking* **15** (1), 40–53, 2007.

[382] Lu, K. *et al.* Robust and Efficient Detection of DDoS Attacks for Large-scale Internet. *Computer Networks* **51**, 5036–5056, 2007.

[383] Shiaeles, S. N. *et al.* Real Time DDoS Detection Using Fuzzy Estimators. *Computers & Security* **31** (6), 782–790, 2012.

[384] Chen, R. *et al.* A Divide-and-Conquer Strategy for Thwarting Distributed Denial-of-Service Attacks. *IEEE Transactions on Parallel and Distributed Systems* **18** (5), 577–588, 2007.

[385] Yaar, A. *et al.* StackPi: New Packet Marking and Filtering Mechanisms for DDoS and IP Spoofing Defense. *IEEE Journal on Selected Areas in Communications* **24** (10), 1853–1863, 2006.

[386] Yoon, M. Using Whitelisting to Mitigate DDoS Attacks on Critical Internet Sites. *IEEE Communications Magazine* **48** (7), 110–115, 2010.

[387] Duan, Z. *et al.* Controlling IP Spoofing Through Interdomain Packet Filters. *IEEE Transactions on Dependable Secure Computing* **5** (1), 22–36, 2008.

[388] Zhang, C. *et al.* Flow Level Detection and Filtering of Low-rate DDoS. *Computer Networks* **56** (15), 3417–3431, 2012.

[389] Shevtekar, A. *et al.* Low Rate TCP Denial-of-Service Attack Detection at Edge Routers. *IEEE Communications Letters* **9** (4), 363–365, 2005.

[390] Gu, Y. *et al.* Detecting Anomalies in Network Traffic Using Maximum Entropy Estimation. In *Proc. of the 5th ACM SIGCOMM conference on Internet Measurement*. 32–32, USENIX Association, Berkeley, CA, USA, 2005.

[391] Martins, A. F. T. *et al.* Nonextensive Entropic Kernels. In *Proc. of the 25th international conference on Machine learning*. 640–647, ACM, New York, NY, USA, 2008.

[392] Mirkovic, J. & Reiher, P. A Taxonomy of DDoS Attack and DDoS Defense Mechanisms. *ACM SIGMETRICS Performance Evaluation Review* **34** (2), 39–53, 2004.

[393] Stein, L. D. & Stewart, J. N. The World Wide WebSecurity FAQ, version 3.1.2. http://www.w3.org/Security/Faq, 2002. Cold Spring Harbor, NY.

[394] Kaspersky. Kaspersky Internet Security & Anti-virus. http://www.kaspersky.com/, 2012. Russian Federation.

[395] Houle, K. J. & Weaver, G. M. Trends in Denial of Service Attack Technology. Tech. Rep. v1.0, CERT and CERT coordination center, Carnegie Mellon University, Pittsburgh, PA, 2001.

[396] Wong, T. Y. *et al.* An Efficient Distributed Algorithm to Identify and Trace-back DDoS Traffic. *Comp. J.* **49** (4), 418–442, 2006.

[397] Collins, J. R. RAMEN - A Linux Worm. http://www.giac.org/paper/gsec/505/ramen-linux-worm/101193, 2000. SANS Institute.

[398] CERT Coordination Center, CERT Advisory CA-2001-19 'Code Red' worm exploiting buffer overflow in IIS indexing service DLL. http://www.cert.org/adviso-ries/CA-2001-19.html, 2001. Carnegie Mellon Software Engineering Institute.

[399] Yu, S. *et al.* Traceback of DDoS Attacks Using Entropy Variations. *IEEE Transactions on Parallel and Distributed Systems* **22** (3), 412–425, 2011.

[400] Chen, Y. *et al.* Distributed Change-Point Detection of DDoS Attacks over Multiple Network Domains. In *Proc. of the IEEE International Symposium on Collaborative Technologies and Systems.* 543–550, IEEE CS, Las Vegas, NV, 2006.

[401] Mirkoviac, J. *et al.* Attacking DDoS at the Source. In *Proc. of the 10th IEEE International Conference on Network Protocols.* 1092–1648, IEEE CS, Washington, DC, USA, 2002.

[402] Saifullah, A. M. Defending Against Distributed Denial-of-Service Attacks with Weight-Fair Router Throttling. Tech. Rep. 2009-7, Computer Science and Engineering, Washington University, St. Louis, USA, 2009.

[403] Feinstein, L. & Schnackenberg, D. Statistical Approaches to DDoS Attack Detection and Response. In *Proc. of the DARPA Information Survivability Conference and Exposition.* 303–314, 2003.

[404] Akella, A. *et al.* Detecting DDoS Attacks on ISP Networks. In *Proc. of the Workshop on Management and Processing of Data Streams.* 1–2, ACM, San Diego, CA, 2003.

[405] Peng, T. *et al.* Detecting Distributed Denial of Service Attacks Using Source IP Address Monitoring. In *Proc of the 3rd International IFIP-TC6 Networking Conference* 771–782, Athens, Greece, 2004

[406] Cheng, J *et al* DDoS Attack Detection Method Based on Linear Prediction Model. In *Proc of the 5th international conference on Emerging intelligent computing technology and applications.* 1004–1013, Springer-Verlag, Ulsan, South Korea, 2009

[407] Udhayan, J & Hamsapriya, T Statistical Segregation Method to Minimize the False Detections During DDoS Attacks. *International Journal of Network Security* **13** (3), 152–160, 2011.

[408] Jalili, R *et al* Detection of Distributed Denial of Service Attacks Using Statistical Pre-Processor and Unsupervised Neural Networks. In *Proc. of the Internationational conference on information security practice and experience* 192–203, Springer-verlag, Singapore, 2005.

[409] Karimazad, R & Faraahi, A An Anomaly-Based Method for DDoS Attacks Detection using RBF Neural Networks. In *Proc. of the International Conference on Network and Electronics Engineering*, vol. 11. 44–48, IACSIT Press, Singapore, 2011

[410] Nguyen, H-V & Choi, Y. Proactive Detection of DDoS Attacks Utilizing k-NN Classifier in an Anti-DDoS Framework. *International Journal of Electrical, Computer, and Systems Engineering* **4** (4), 247–252, 2010.

[411] Kumar, P. A. R. & Selvakumar, S. Distributed Denial of Service Attack Detection Using an Ensemble of Neural Classifier *Computer Communication* **34** (11), 1328–1341, 2011

[412] Scott, C. & Nowak, R. A Neyman-Pearson Approach to Statistical Learning *IEEE Transactions on Information Theory* **51** (11), 3806–3819, 2005.

[413] Gil, T. M. & Poletto, M MULTOPS: A Data-structure for Bandwidth Attack Detection. In *Proc. of the 10th conference on USENIX Security Symposium - Volume 10* 3–3, USENIX Association Berkeley, Berkeley, CA, USA, 2001.

[414] Thomas, R. *et al.* NetBouncer: Client-legitimacy-based High-performance DDoS Filtering. In *Proc. of the 3rd DARPA Information Survivability Conference and Exposition* 111–113, IEEE CS, USA, Washington, DC, 2003

[415] Wang, J. *et al.* Augmented Attack Tree Modeling of Distributed Denial of Services and Tree Based Attack Detection Method. In *Proc. of the 10th IEEE International Conference on Computer and Information Technology.* 1009–1014. IEEE CS, Bradford, UK, 2010

[416] Limwiwatkul, L. & Rungsawang, A. Distributed Denial of Service Detection using TCP/IP Header and Traffic Measurement Analysis. In *Proc. of the IEEE International Symposium Communications and Information Technology.* 605–610, IEEE CS, Sapporo, Japan, 2004.

[417] Hwang, K. *et al.* NetShield: Protocol Anomaly Detection with Datamining Against DDoS Attacks. In *Proc. of the 6th International Symposium on Recent Advances in Intrusion Detection.* 8–10, Springer-verlag, Pittsburgh, PA, 2003.

[418] Lee, K. *et al.* DDoS Attack Detection Method Using Cluster Analysis. *Expert Syst. Appl.* **34** (3), 1659–1665, 2008

[419] Sekar, V. *et al.* LADS: Large-scale Automated DDoS Detection System. In *Proc. of the annual conference on USENIX Annual Technical Conference.* 16–29, USENIX Association, Boston, MA, 2006

[420] Dainotti, A. *et al* A Cascade Architecture for DoS Attacks Detection Based on the Wavelet Transform. *Journal of Computer Security* **17** (6), 945–968, 2009.

[421] Haar, A. Zur Theorie Der Orthogonalen Funktionensysteme. *Mathematische Annalen* **69** (3), 331–371. 1910.

[422] Li, L. & Lee, G. DDoS Attack Detection and Wavelets. In *Proc. of the 12th International Conference on Computer Communications and Networks.* 421–427, IEEE, Dallas, Texas, USA, 2003.

[423] Xia, Z. *et al.* Enhancing DDoS Flood Attack Detection via Intelligent Fuzzy Logic. *Informatica (Slovenia)* **34** (4). 497–507, 2010.

[424] Gelenbe, E. & Loukas, G. A Self-aware Approach to Denial of Service Defence. *Computer Networks* **51** (5), 1299–1314, 2007.

[425] Yuan, J. & Mills, K. Monitoring the Macroscopic Effect of DDoS Flooding Attacks. *IEEE Transactions on Dependable Secure Computing* **2** (4), 324–335, 2005.

[426] Shannon, C. E. A Mathematical Theory of Communication. *Bell system technical journal* **27**, 397–423, 1948.

[427] Francois, J. *et al.* FireCol: A Collaborative Protection Network for the Detection of Flooding DDoS Attacks. *IEEE/ACM Transaction on Networking* **20** (6), 1828–1841, 2012.

[428] Wei, W. *et al.* A Rank Correlation Based Detection against Distributed Reflection DoS Attacks. *IEEE Communications Letters* **17** (1), 173–175, 2013.

[429] Bhatia, S. *et al.* Ensemble-based DDoS Detection and Mitigation Model. In *Proc. of the Fifth International Conference on Security of Information and Networks*. 79–86, ACM, Jaipur, India, 2012.

[430] Masi, M. A Step Beyond Tsallis and Renyi Entropies. *Physics Letters A* **338** (3-5), 217–224, 2005.

[431] Rényi, A. On Measures of Entropy And Information. In *Proc. of the 4th Berkeley Symposium on Mathematics, Statistics and Probability*. 547–561, 1960.

[432] Douligeris, C. & Mitrokotsa, A. DDoS Attacks and Defense Mechanisms: Classification and State-of-the-art. *Computer Networks* **44** (5), 643–666, 2004.

[433] Criscuolo, P. J. Distributed Denial of Service Trinoo, Tribe Flood Network, Tribe Flood Network 2000. Rev. 1 UCRL-ID-136939, Department of Energy Computer Incident Advisory (CIAC), Lawrence Livermore National Laboratory, Livermore, CA, http://ftp.se.kde.org/pub/security/csir/ciac/ciacdocs/ciac2319.txt, 2000.

[434] Dittrich, D. The DoS Project's "trinoo" Distributed Denial of Service Attack Tool. Tech. Rep., University of Washington, Seatlle, USA, http://staff.washington.edu/dittrich/misc/trinoo analysis.txt, 1999.

[435] Dittrich, D. The Tribe Flood Network Distributed Denial of Service Attack Tool. Tech. Rep., University of Washington, Seatlle, USA, http://staff.washington.edu/dittrich/misc/trinoo analysis.txt, 1999.

[436] Barlow, J. & Thrower, W. TFN2K ? an Analysis. http://packetstormsecurity.org/files/10135/TFN2k_Analysis-1 3.txt.html, 2000. AXENT Security Team

[437] CERT. CERT Coordination Center, Center Advisory CA-1999-17 denial of service tools. Tech Rep.

[438] Adams, C. & Gilchrist, J. The CAST-256 Encryption Algorithm. Tech. Rep. RFC 2612, Ohio State University, Cleveland, USA, http://www.cis.ohio-state.edu/htbin/rfc/rfc2612.html, 1999.

[439] Bellovin, S. The ICMP Traceback Message. Tech. Rep., Network Working Group, Internet Draft, Canada, http://www.research.att com/ smb/papers/draft-bellovin-itrace-00.txt, 2000.

[440] Dittrich, D. The 'Stacheldraht' Distributed Denial of Service Attack Tool. Tech. Rep., University of Washington, Seatlle, USA, http://staff.washington.edu/dittrich/misc/ stacheldraht.analysis.txt, 1999.

[441] Dittrich, D. *et al.* The 'mstream' Distributed Denial of Service Attack Tool. Tech. Rep., University of Washington, Seatlle, USA, http://staff.washington.edu/dittrich/misc/mstream.analysis.txt, 2000.

[442] Dietrich, S. *et al.* Analyzing Distributed Denial of Service Tools: The Shaft Case. In *Proc. of the 14th USENIX conference on System administration.* 329–340, USENIX Association, Berkeley, USA, 2000.

[443] Hancock, B. Trinity v3, a DDoS Tool, Hits the Streets. *Computers & Security* **19** (7), 574, 2000.

[444] King, B. B. & Morda, D. CERT Coordination Center, CERT Advisory CA-2001-20 Continuing Threats to Home Users. Tech. Rep. CA-2001-20, Carnegie Mellon Software Engineering Institute, http://www.cert.org/advisories/CA-2001-20.html, 2001.

[445] Bysin. Knight.c Source Code, PacketStormSecurity.nl. http://packetstormsecurity.nl/distributed/knight.c, 2001.

[446] MIT Lincoln Laboratory Datasets. MIT LLS_DDOS_0.2.2. http://www.ll.mit.edu/mission/communications/cyber/CSTcorpora/ideval/data/2000data.html, 2000, Massachusetts Institute of Technology, Cambridge, MA.

[447] Chen, Y. *et al.* DDoS Detection Algorithm Based on Preprocessing Network Traffic Predicted Method and Chaos Theory. *IEEE Communications Letters* **17** (5), 1052–1054, 2013.

List of publications

(a) Refereed journal papers

1. **M. H. Bhuyan**, D. K. Bhattacharyya and J. K. Kalita. **Survey on Incremental Approaches for Network Anomaly Detection**, in *International Journal of Communication Networks and Information Security*, vol. 3, no. 3, pp. 226–239, 2011.

2. **M. H. Bhuyan**, D. K. Bhattacharyya and J. K. Kalita. **Surveying Port Scans and Their Detection Methodologies**, in *The Computer Journal*, British Computer Society, Oxford University Press, vol. 54, no. 10, pp. 1565–1581, 2011, UK. (**SCI, Impact Factor: 1.39**)

3. **M. H. Bhuyan**, D. K. Bhattacharyya and J. K. Kalita. **AOCD: An Adaptive Outlier Based Coordinated Scan Detection Approach**, in *International Journal of Network Security*, vol. 14, no. 6, pp. 339–351, 2012.

4. **M. H. Bhuyan**, H. Kashyap, D. K. Bhattacharyya and J. K. Kalita. **Distributed Denial of Service Attack Detection : Methods, Tools and Future Directions**, in *The Computer Journal*, British Computer Society, Oxford University Press, DOI: 10.1093/comjnl/bxt031, 2013, UK (**SCI, Impact Factor: 0.79**).

5. **M. H. Bhuyan**, D. K. Bhattacharyya and J. K. Kalita. **Towards an Unsupervised Method for Network Anomaly Detection in Large Datasets**, in *Journal of Computing and Informatics*, vol. 33, no. 1, 2014 (**in press, SCI, Impact Factor: 0.24**).

6. **M. H. Bhuyan**, D. K. Bhattacharyya and J. K. Kalita. **An Efficient Outlier Based Technique for Analysis of High Dimensional and Large Network Traffic Datasets**, in *Pattern Recognition*, Elsevier, 2012 (**revision submitted, SCI, Impact Factor: 2.63**)

7. **M. H. Bhuyan**, D. K. Bhattacharyya and J. K. Kalita. **Network Anomaly Detection : Methods, Systems and Tools**, in *IEEE Communications Surveys and Tutorials*, DOI: 10.1109/SURV.2013.052213.00046, 2013, USA, **(SCI, Impact Factor: 6.31)**.

8. N. Hoque **M. H. Bhuyan**, R. C. Baishya, D. K. Bhattacharyya and J. K. Kalita. **Network Attacks: Taxonomy, Tools and Systems**, in *Journal of Network and Computer Applications*, DOI: 10.1016/j.jnca.2013.08.001, Elsevier, 2013, USA **(accepted, SCI, Impact Factor: 1.47)**.

9. **M. H. Bhuyan**, D. K. Bhattacharyya and J. K. Kalita. **DDoS Flooding Attack Detection and IP Traceback Using Extended Entropy Metric**, in *IEEE Transactions on Parallel and Distributed Systems*, IEEE Computer Society, USA, 2013 **(submitted, SCI, Impact Factor: 1.41)**.

10. **M. H. Bhuyan**, D. K. Bhattacharyya and J. K. Kalita. **Towards a Systematic Approach to Generate Real-life Datasets for Intrusion Detection**, in *Journal of Computer Standards and Interfaces*, Elsevier, 2013, USA **(submitted, SCI, Impact Factor: 0.98)**.

11. **M. H. Bhuyan**, D. K. Bhattacharyya and J. K. Kalita. **An Empirical Evaluation of Information Metrics for Detecting Low-rate and High-rate DDoS Attacks**, in *Pattern Recognition Letters*, Elsevier, 2014, USA **(submitted, SCI, Impact Factor: 1.27)**.

(b) Refereed conference papers

1. **M. H. Bhuyan**, D. K. Bhattacharyya and J. K. Kalita. **RODD: An Effective Reference based Outlier Detection Technique for Large Datasets**, in *Advanced Computing*, CCSIT, Bangalore, India, Jan 2-4, LNCS-CCIS, Vol. 133, Part III, pp. 76–84, 2011, Springer-Verlag.

2. **M. H. Bhuyan**, D. K. Bhattacharyya and J. K. Kalita. **NADO : Network Anomaly Detection Using Outlier Approach**, in *Proceedings of the International Conference on Communication, Computing and Security*, Feb 12-14, Rourkela, Odisha, India, pp. 531–536, 2011, ACM.

3. **M. H. Bhuyan**, D. K. Bhattacharyya and J. K. Kalita. **An Effective Unsupervised Network Anomaly Detection Method**, in *Proceedings of the International Conference on Advances in Computing, Communications and Informatics*, Aug 3-5, Chennai, India, pp. 533–539, 2012, ACM. (**Best Paper Award**)

4. P. Gogoi, **M. H. Bhuyan**, D. K. Bhattacharyya and J. K. Kalita. **Packet and Flow Based Network Intrusion Datasets**, in *Proceedings of the International Conference on Contemporary Computing*, Aug 6-8, Noida, India, LNCS-CCIS Vol. 306, pp. 322–334, 2012, Springer-Verlag.

5. **M. H. Bhuyan**, D. K. Bhattacharyya and J. K. Kalita. **Information Metrics for Low-rate DDoS Attack Detection : A Comparative Evaluation**, in *SIAM SDM-OMAD 2014*.